Как сдержать самые сложные обещания: планирование batch-задач в системе Yandex.YT

Колесниченко Игнат (Яндекс)





Содержание

01

Введение, или зачем мы строим большие кластеры YT

Жизнь в больших компаниях



Жизнь в больших компаниях

- Сотня подразделений со своими задачами



Жизнь в больших компаниях

- Сотня подразделений со своими задачами
- Общие данные



Жизнь в больших компаниях

- Сотня подразделений со своими задачами
- Общие данные
- Куча железа

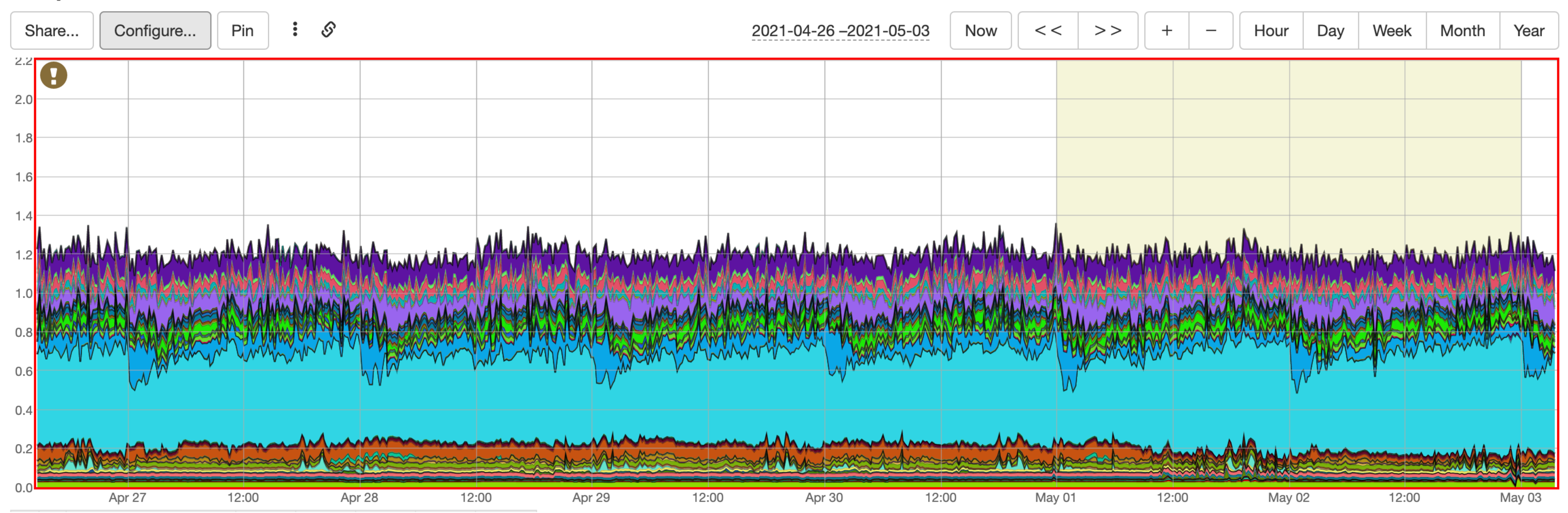


Жизнь в больших компаниях


- Сотня подразделений со своими задачами
- Общие данные
- Куча железа
- Непостоянное использование вычислительных ресурсов



Доля используемых ресурсов во времени

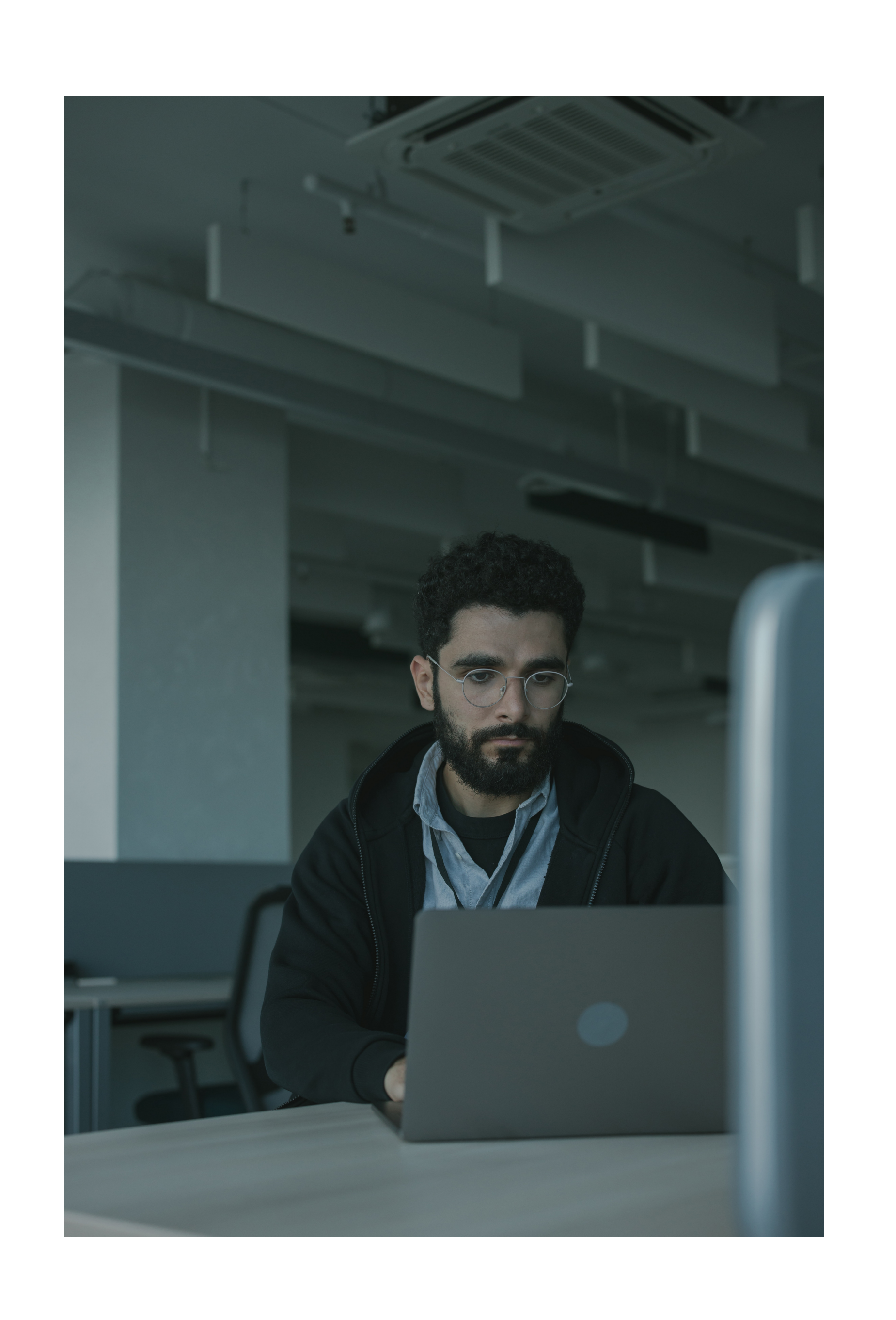


Зачем нам один общий кластер?



Свой кластер – своя эксплуатация

- В Яндексе 50+ крупных подразделений
- Свой кластер – своя команда SRE



x 50 vs



~30 SWE

Что такое YT

- Внутренняя система для хранения и обработки данных Яндекса
- ~30 разработчиков в отделе
- 30'000+ серверов в эксплуатации
- 1'000'000+ CPU на самом крупном кластере

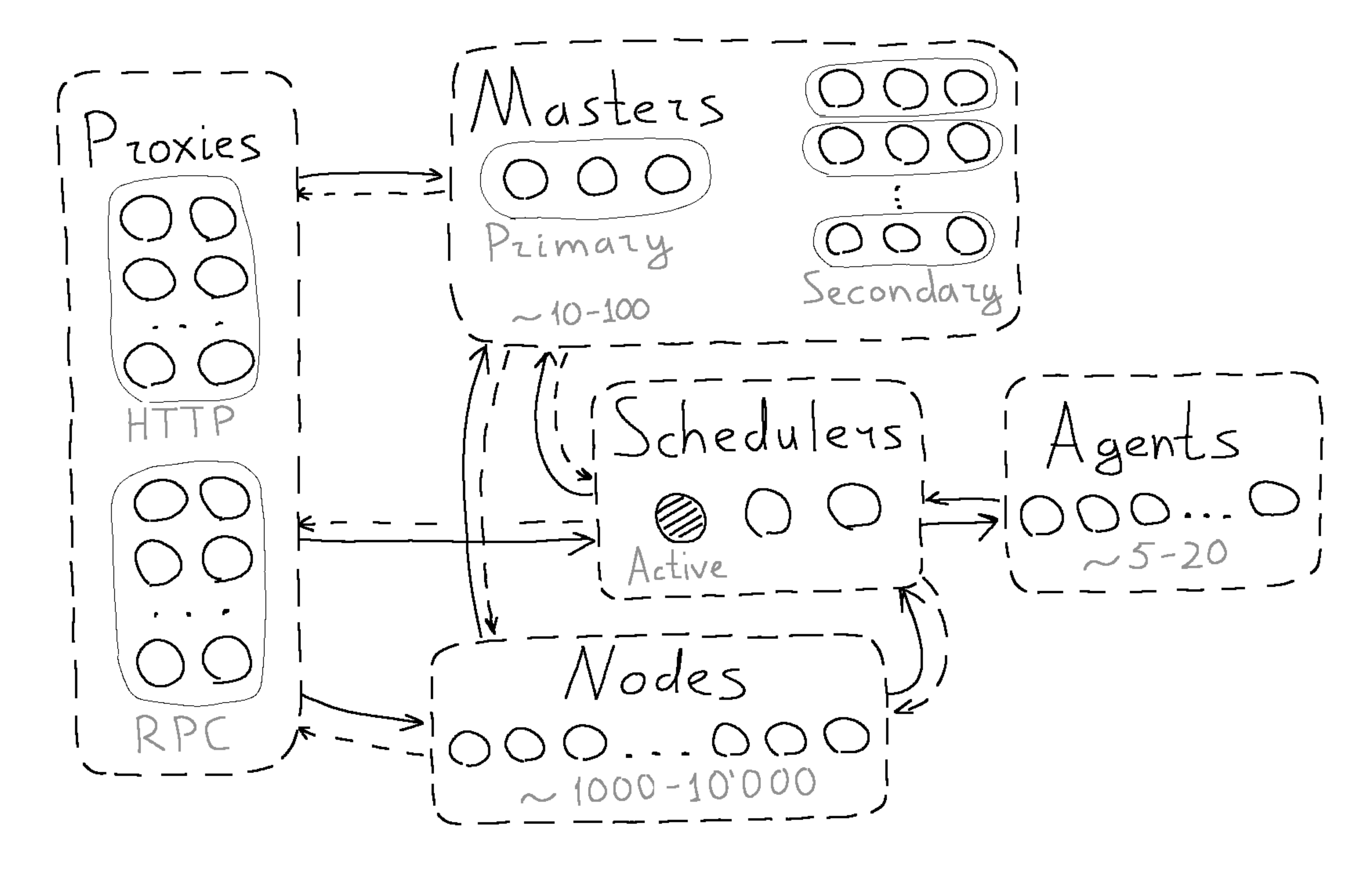
Чем занимаются кластеры YT

- Хранение метаинформации (Masters)
- Хранение больших данных (Nodes)
- Обработка данных
  - Планирование (Schedulers and Agents)
  - Исполнение (Nodes)
- OLTP KV-хранилище (Tablet Nodes)

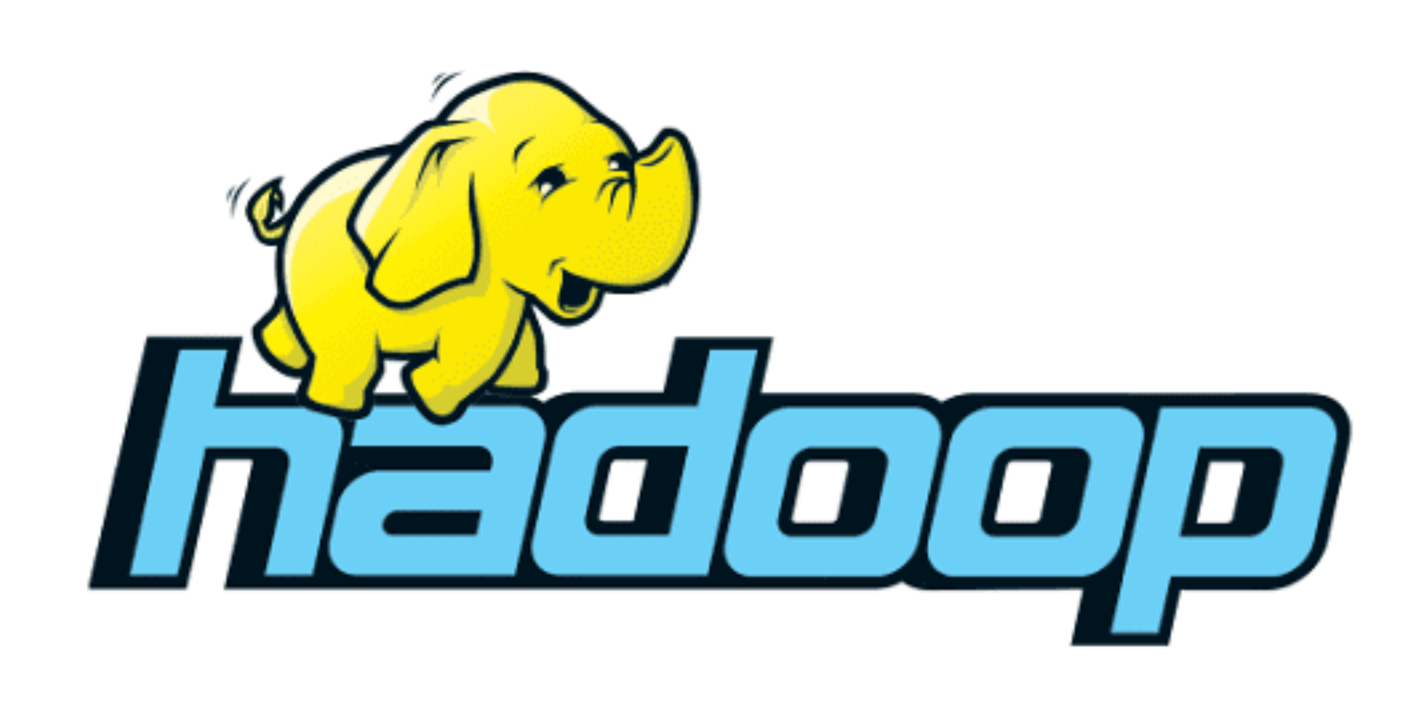
Hadoop Universe

- NameNode, Zookeeper
- DataNode
- Hadoop MapReduce
  - Yarn and App Masters
  - NodeManager
- HBase, Cassandra

Условная схема кластера

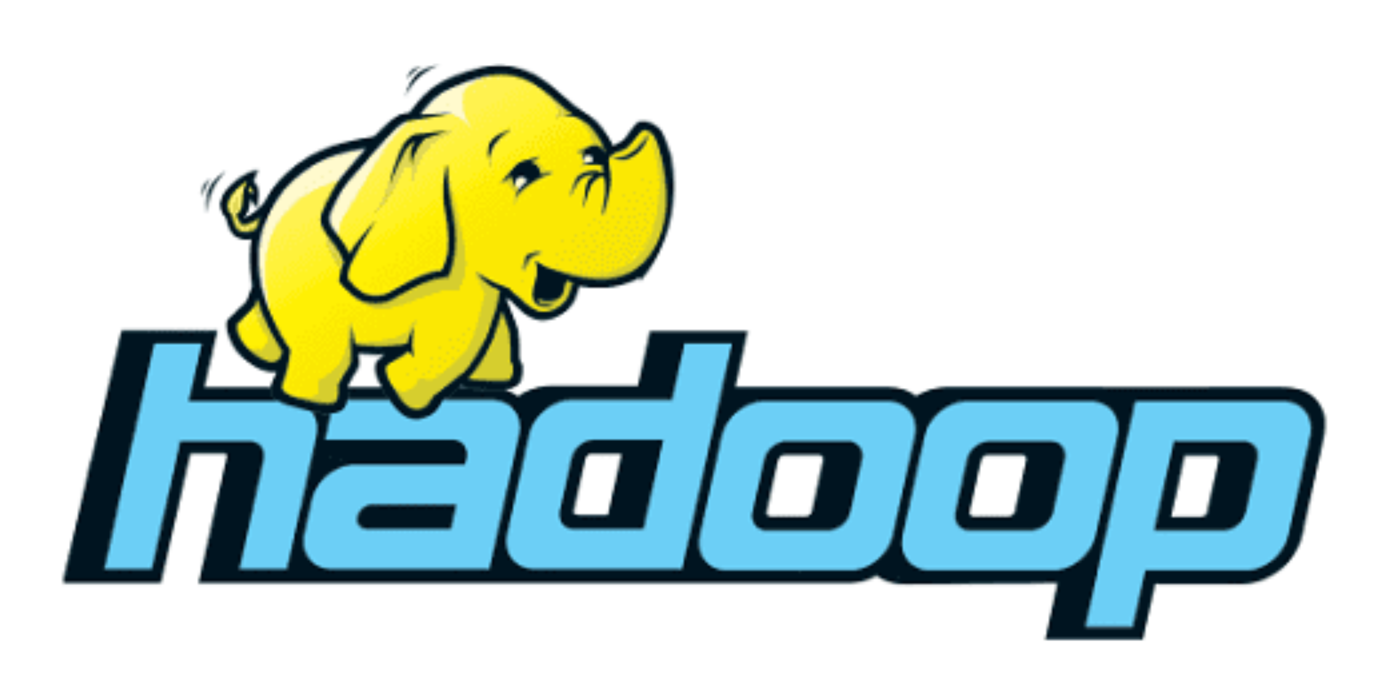


А почему не Hadoop?



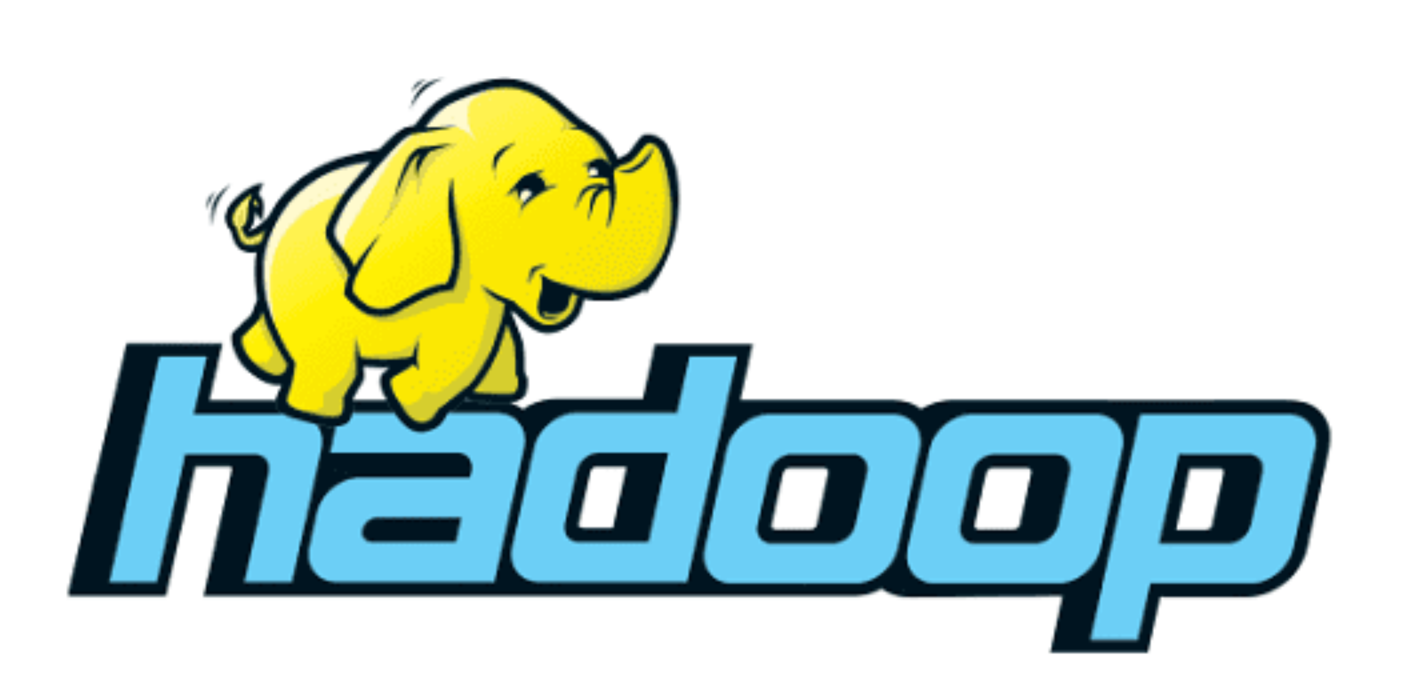
А почему не Hadoop?

Не умеет в наши масштабы



А почему не Hadoop?

- Не умеет в наши масштабы
- Сложная кодовая база



А почему не Hadoop?

- Не умеет в наши масштабы
- Сложная кодовая база
- Мы умеем
  - Соблюдать строгие мгновенные гарантии
  - Обеспечивать интегральные гарантии
  - Решать задачи фрагментации
  - ...



02

Планировщик, кто он такой?

Терминология

Disclaimer!

YT Operation	⟷	Hadoop Job
YT Job	⟷	Hadoop Task

С чем приходится иметь дело планировщику

Исполнение запросов пользователей

```
{
    "input_tables": [...],
    "output_tables": [...],
    "mapper": {
        "command": "./my_wonderful_binary",
        "files": ["//home/babenko/my_wonderful_binary", ...],
        "cpu_guarantee": 10,
        "memory_guarantee": 20'000'000'000,
        ...
    }
    "pool": "dev",
    ...
}
```

С чем приходится иметь дело планировщику

Общение с нодами

Запрос от ноды
- Бегущие джобы
- Свободные ресурсы

Ответ ноде
- Новые джобы
- Аборты некоторых джобов

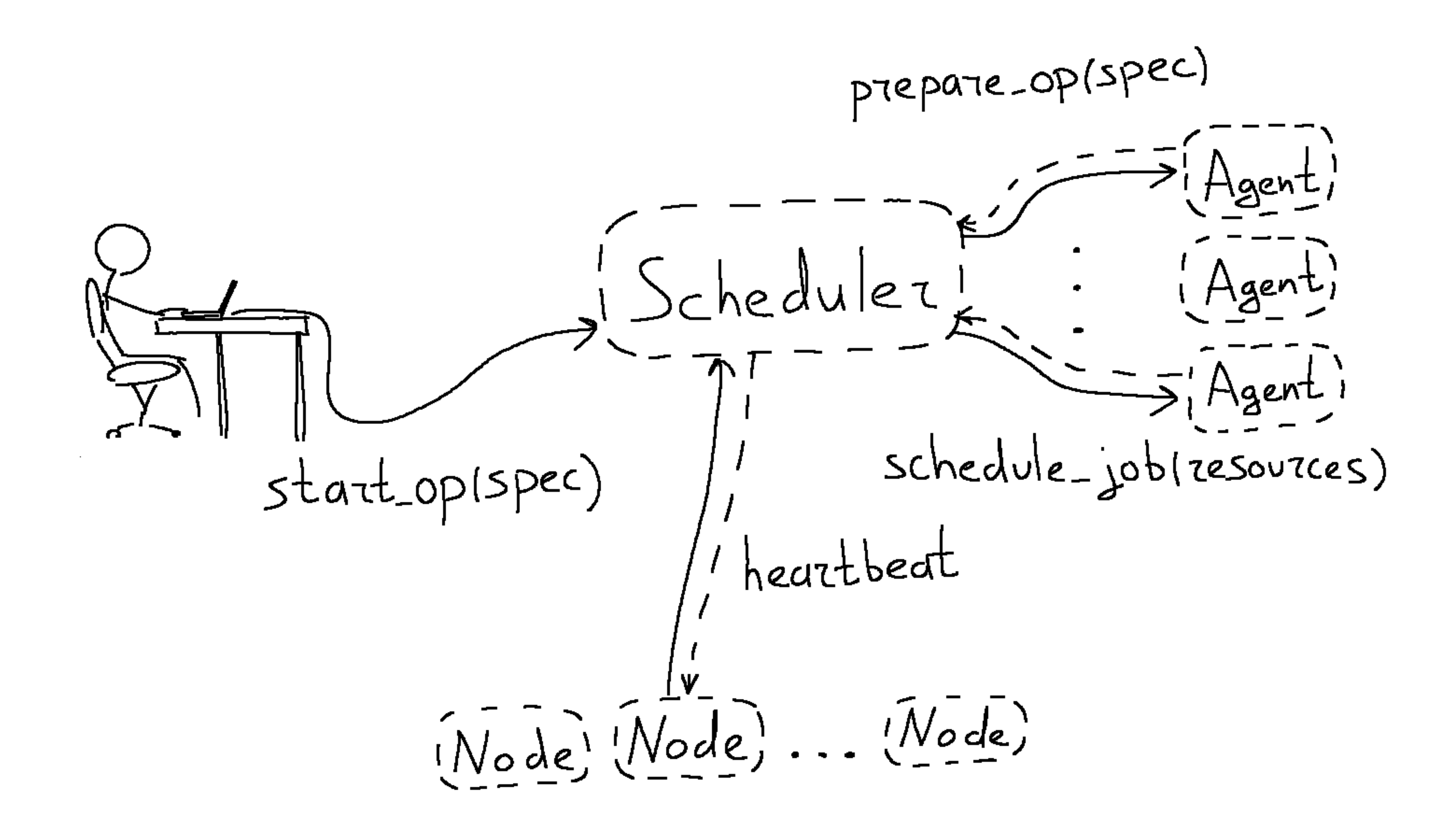
С чем приходится иметь дело планировщику

Общения с агентами, которые планируют операции

Запросы от планировщика
- Сообщает спеку операции
- Сообщает о событиях с джобами
- Просит запланировать джоб


Ответы планировщику
- Запланированные джобы
- Сообщения об окончании операции

С чем приходится иметь дело планировщику



Как тяжело жить

16'000 хартбитов в секунду



Немного статистики

- ~7'000'000 операций в день
- ~7000 джобов завершается за 1 секунду
- ~600'000 попыток предложить ресурсы операции за 1 секунду

03

Так как же поделить ~~торт~~ кластер

Что известно планировщику

x 7000

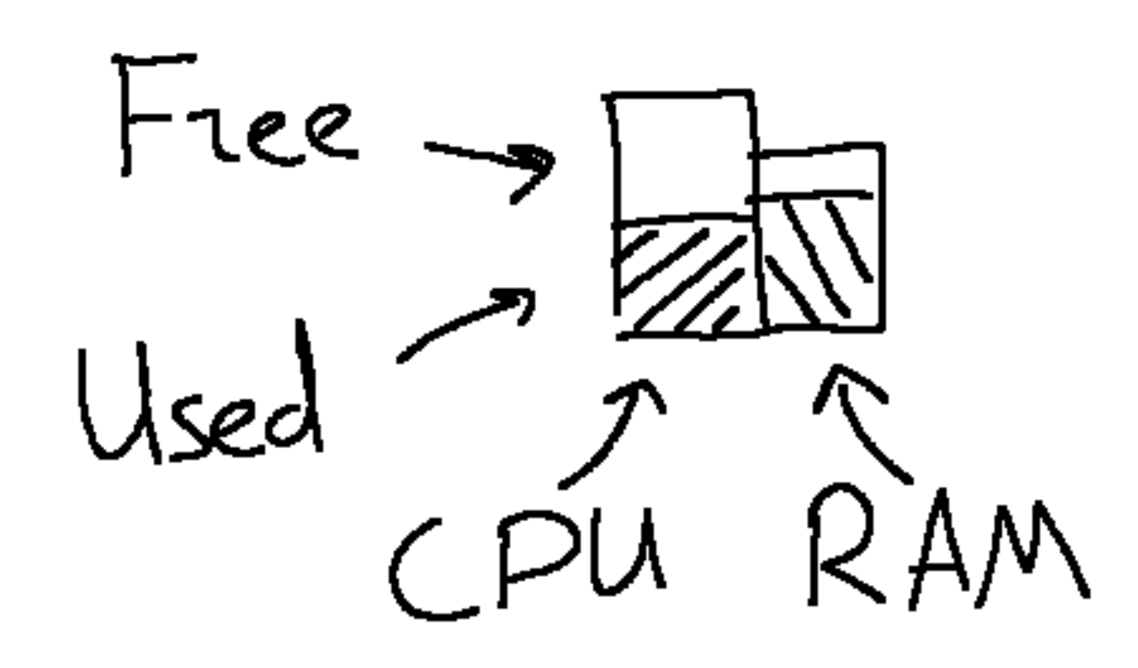
OP_1

x2000 x3000

OP_2

Nodes

Профиль ресурсов



Модель принятия решений

Модель принятия решений

Вводные

- В кластере больше 10'000 нод
- Средняя длительность джоба ~1 минута

Модель принятия решений

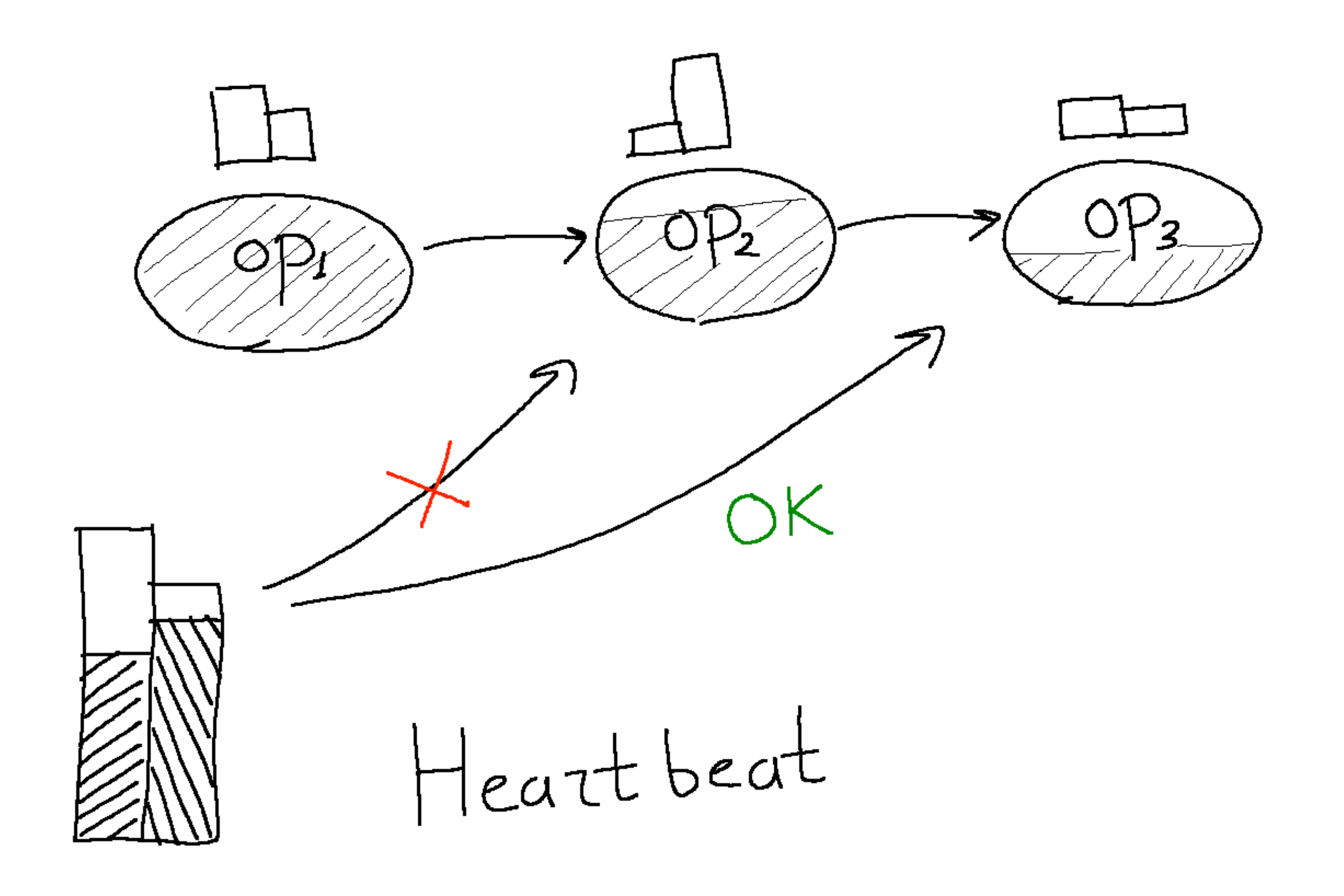
Вводные

- В кластере больше 10'000 нод
- Средняя длительность джоба ~1 минута

Следствие

- Планировать надо очень **быстро** и очень **много**
- Решения о планировании принимаются на каждый хартбит

FIFO-стратегия



Операция

- usage – u_{op}
- demand – d_{op}
- job resources – j_{op}

Нода

- free – r_{node}

Ищем операцию с

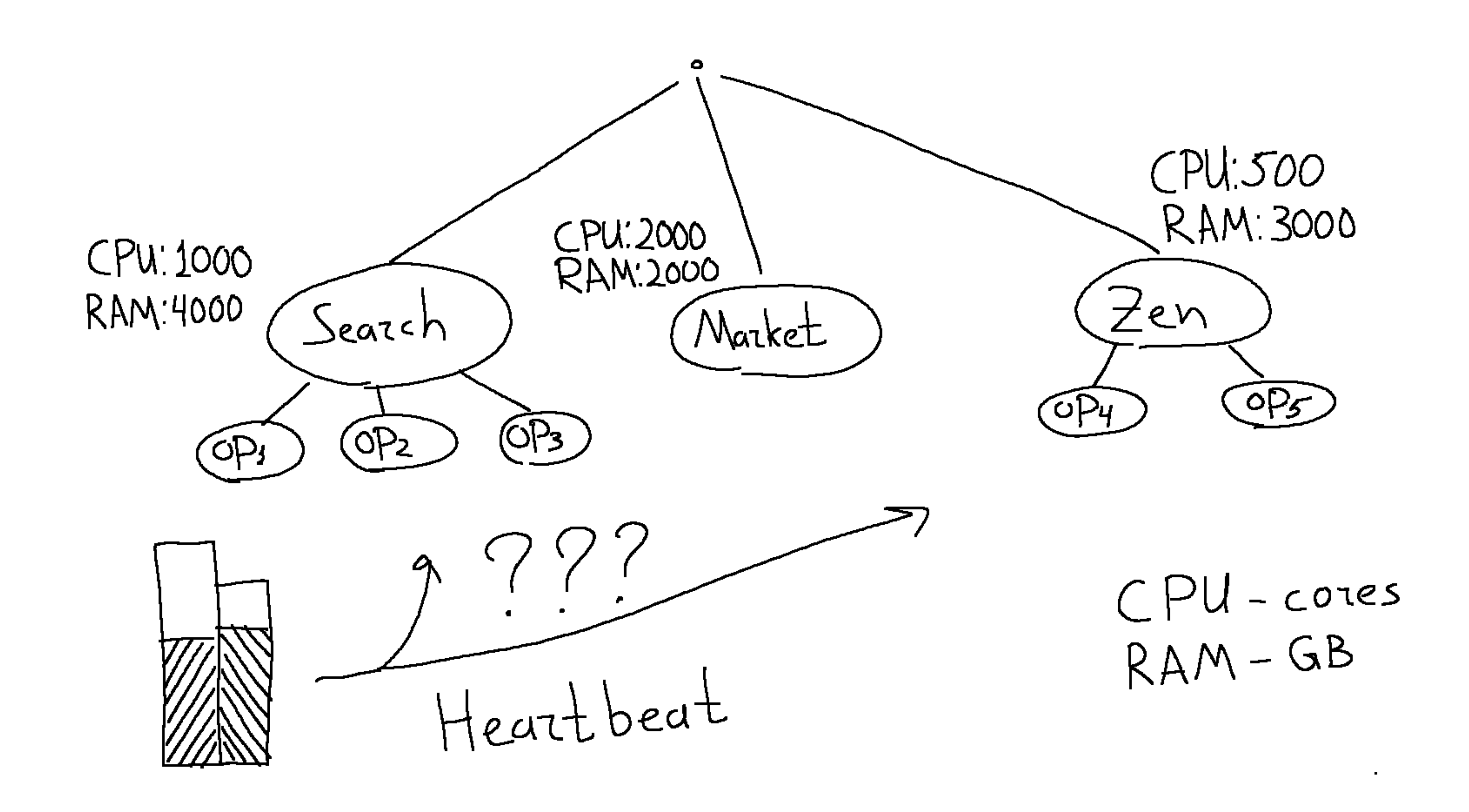
$u_{op} < d_{op}$ и $j_{op} \leq r_{node}$

А как же честность?

Аналитик, запускающий греп логов за год и занимающий весь кластер



Пулы


CPU:1000
RAM:4000
Search
CPU:2000
RAM:2000
Market
CPU:500
RAM:3000
Zen
OP1
OP2
OP3
OP4
OP5
???
Heartbeat
CPU - cores
RAM - GB

Стратегия на основе fairness

Посчитаем величины:

- fs_{pool} – доля кластера, **положенная** пулу
- us_{pool} – доля кластера, **выданная** пулу

Выберем пул с минимальным us_{pool}/fs_{pool}

Max-min fairness

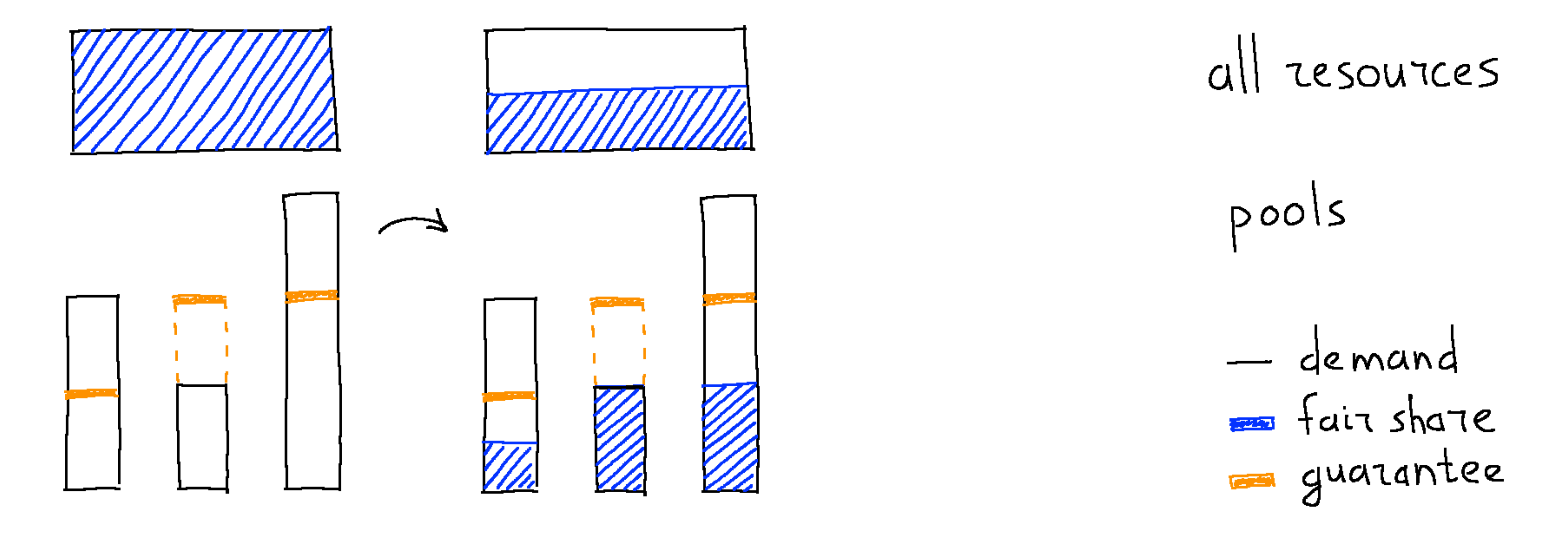
Будем делить **только CPU**, наливаем пропорционально гарантиям*



* Фактически мы делаем бинпоиск по раздаваемой доле ресурсов

Max-min fairness

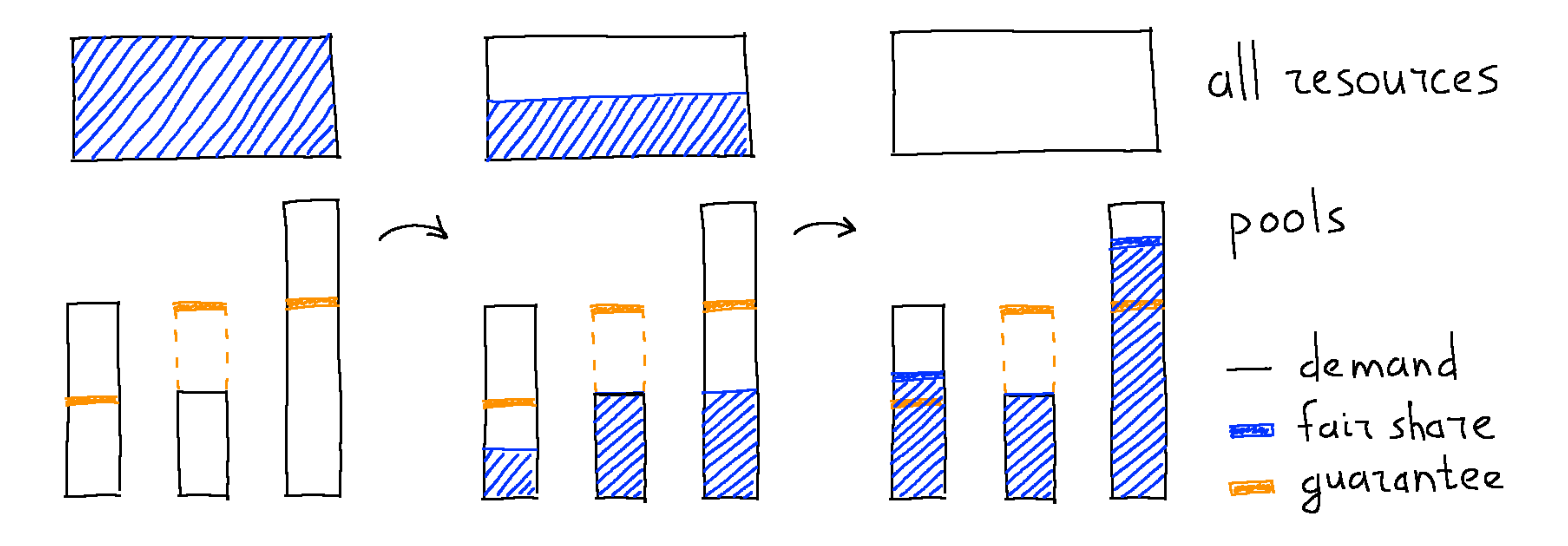
Будем делить **только CPU**, наливаем пропорционально гарантиям*



* Фактически мы делаем бинпоиск по раздаваемой доле ресурсов

Max-min fairness

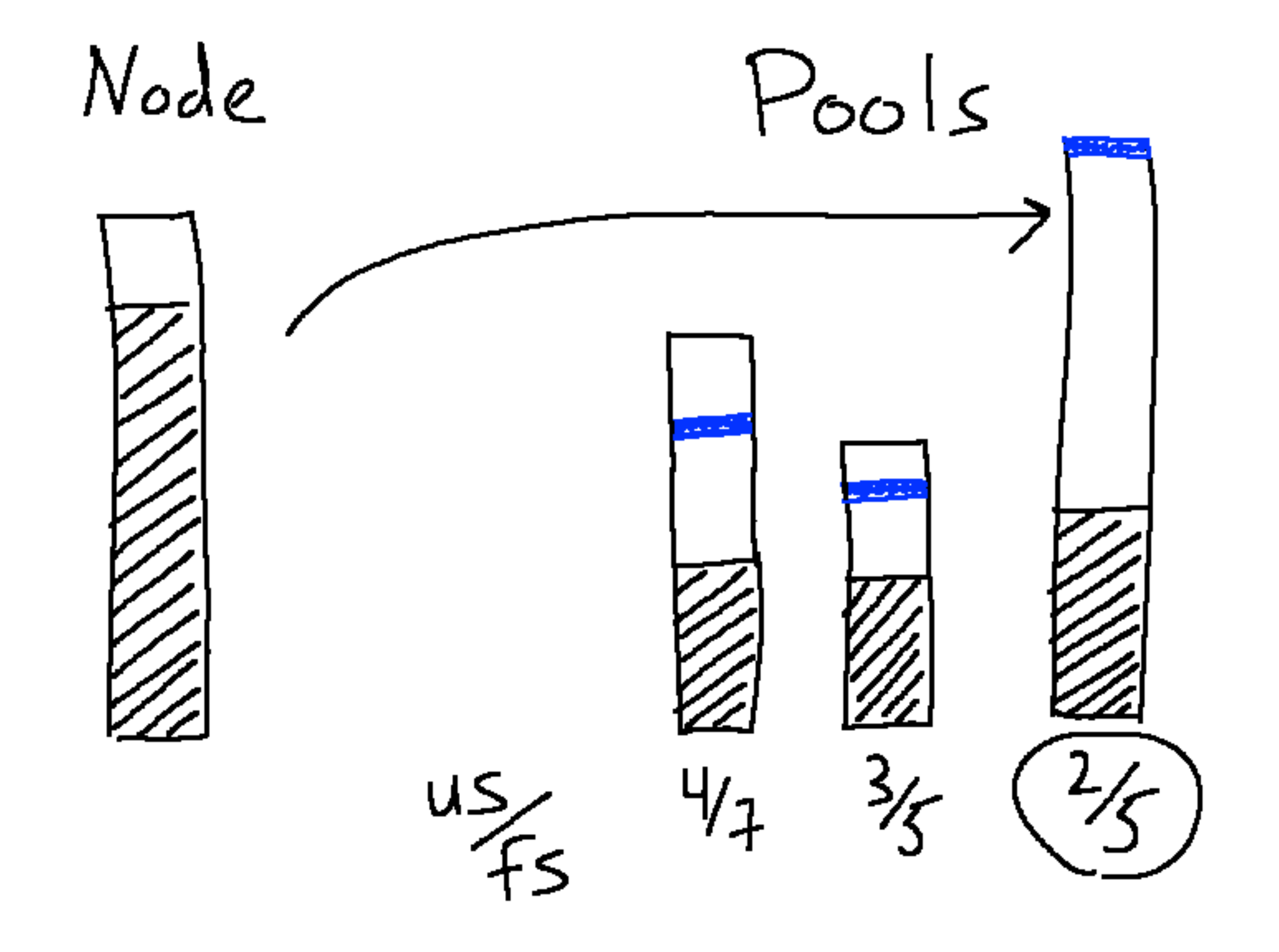
Будем делить **только CPU**, наливаем пропорционально гарантиям*



* Фактически мы делаем бинпоиск по раздаваемой доле ресурсов

Max-min fairness: heartbeat

Выбираем пул с минимальным us_{pool}/fs_{pool}

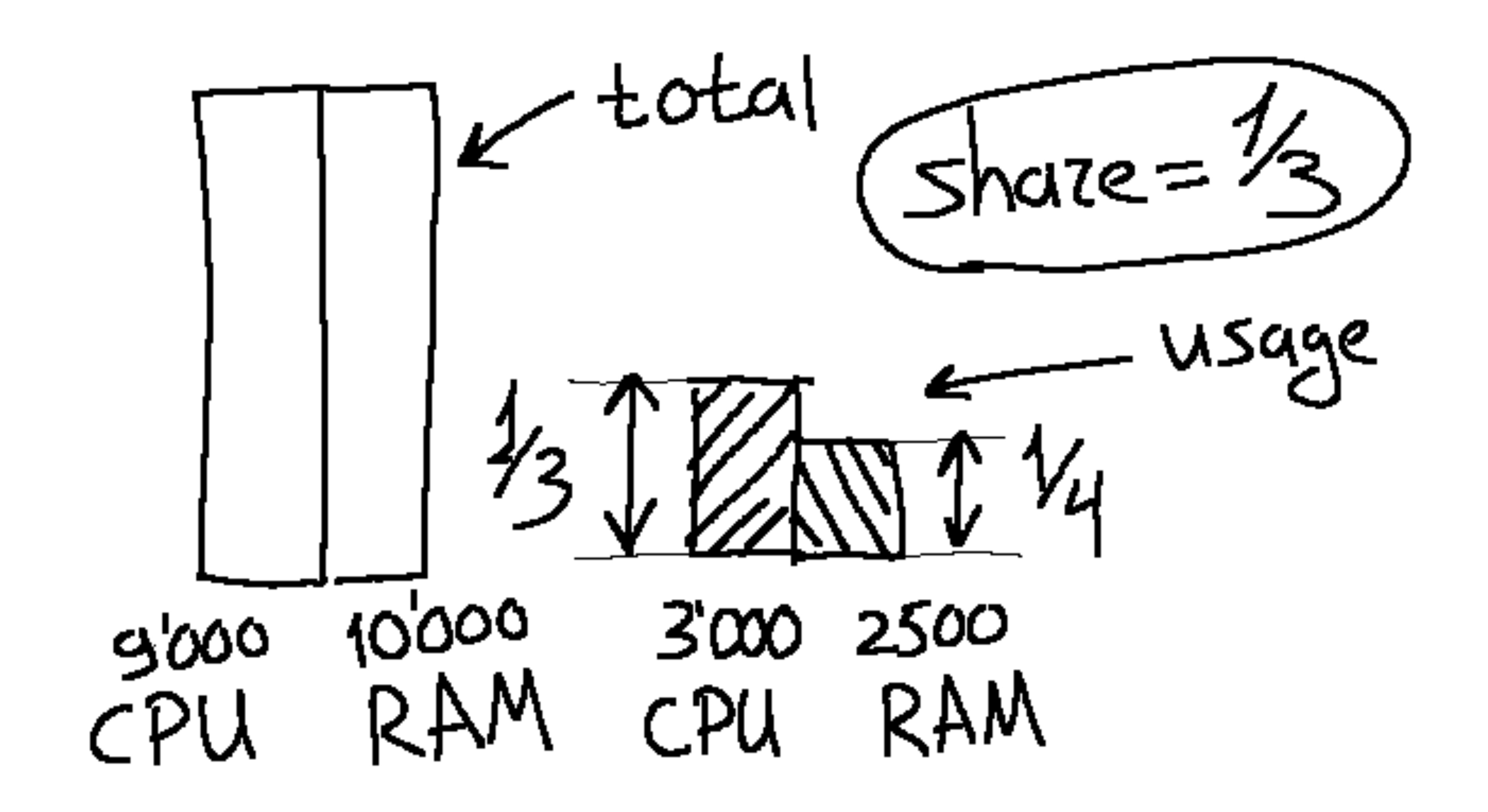


- usage
- fair share

Dominant resource fairness*

Хотим делить **не только CPU**

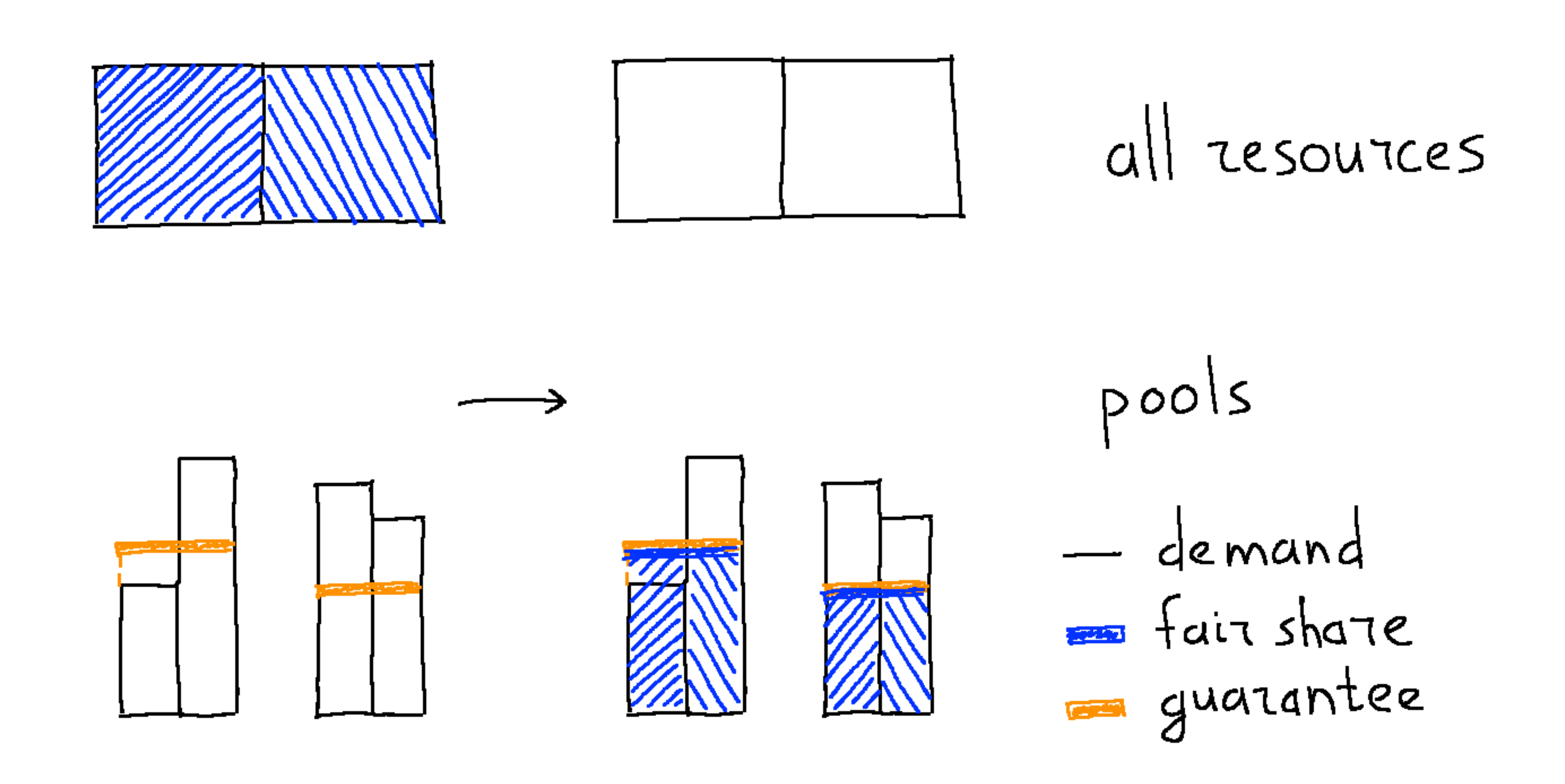
Идея: переведем вектор в число (доминантную долю)



* A.Ghodsi, et.al [2011]: Dominant Resource Fairness: Fair Allocation of Multiple Resource Types

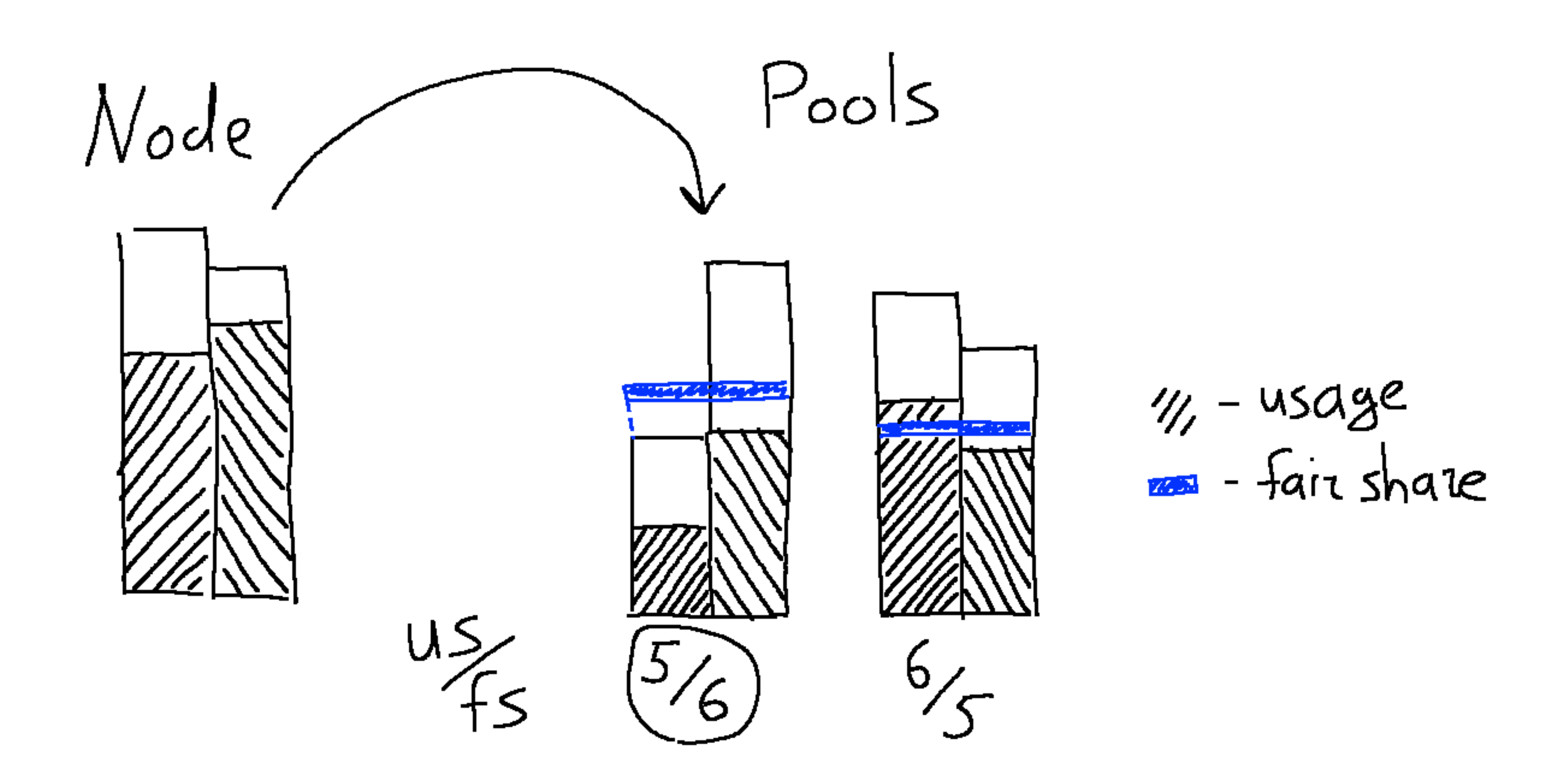
DRF: распределение fair share

Делим все ресурсы, но с **одинаковой** пропорциональностью


DRF: heartbeat

Выбираем пул с минимальным us_{pool}/fs_{pool} (share по доминантному ресурсу)



DRF: насыщение



DRF: свойства

DRF: свойства

- Умеет доедать ресурсы кластера

DRF: свойства

- Умеет доедать ресурсы кластера

- Поддерживает CPU, RAM

DRF: свойства

- Умеет доедать ресурсы кластера
- Поддерживает CPU, RAM
- Дает гарантии по доминантному ресурсу

DRF: свойства

- Умеет доедать ресурсы кластера
- Поддерживает CPU, RAM
- Дает гарантии по доминантному ресурсу
- Излишки раздает пропорционально гарантиям

Чего не хватает?

Хочется делить ресурсы **иерархически**

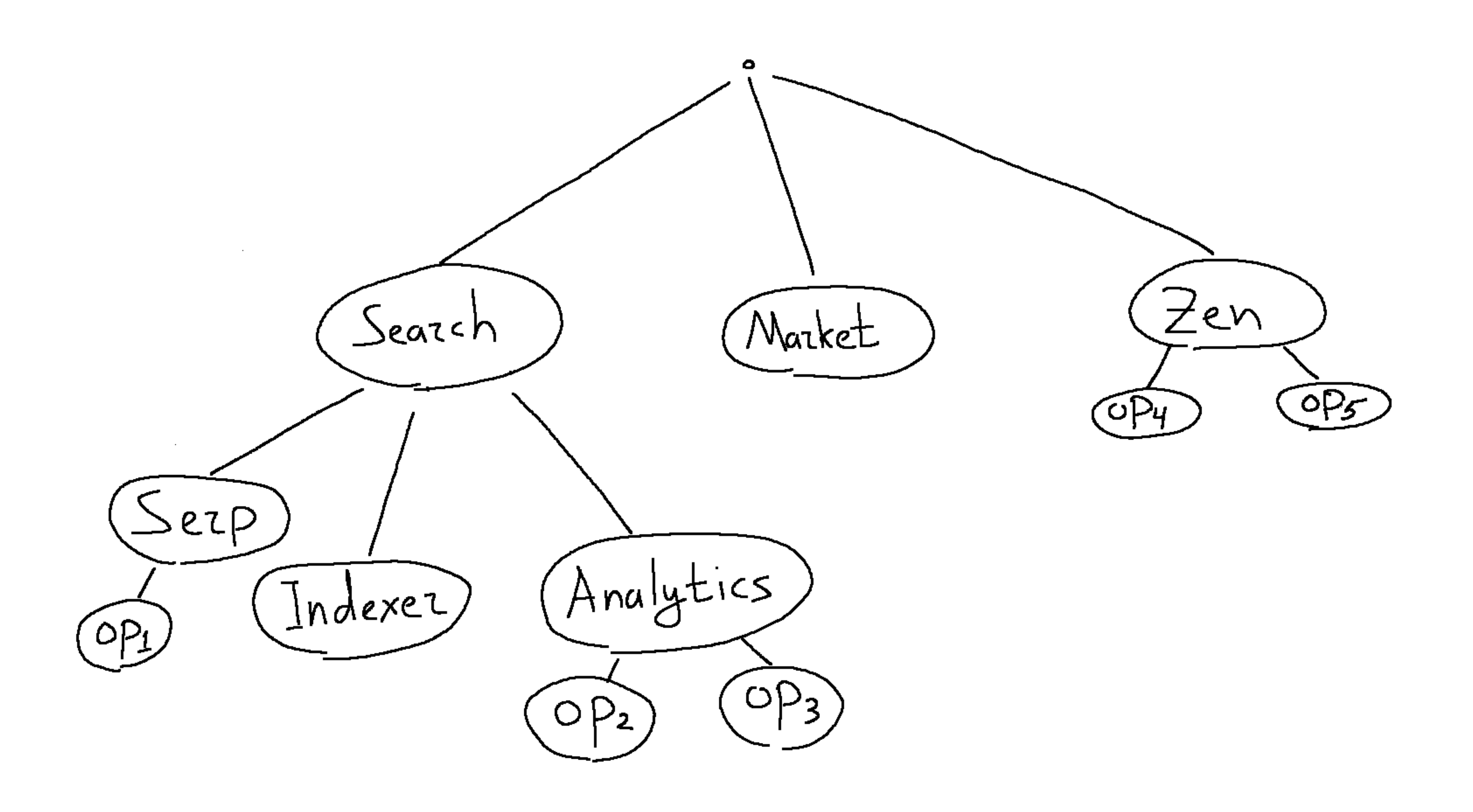
Компания очень большая

- Яндекс.Поиск – 2000+ сотрудников
- Яндекс.Гео – 500+ сотрудников
- ...

```
$ yt list //sys/pools | wc -l
171
$ yt find //sys/pools | wc -l
2170
```

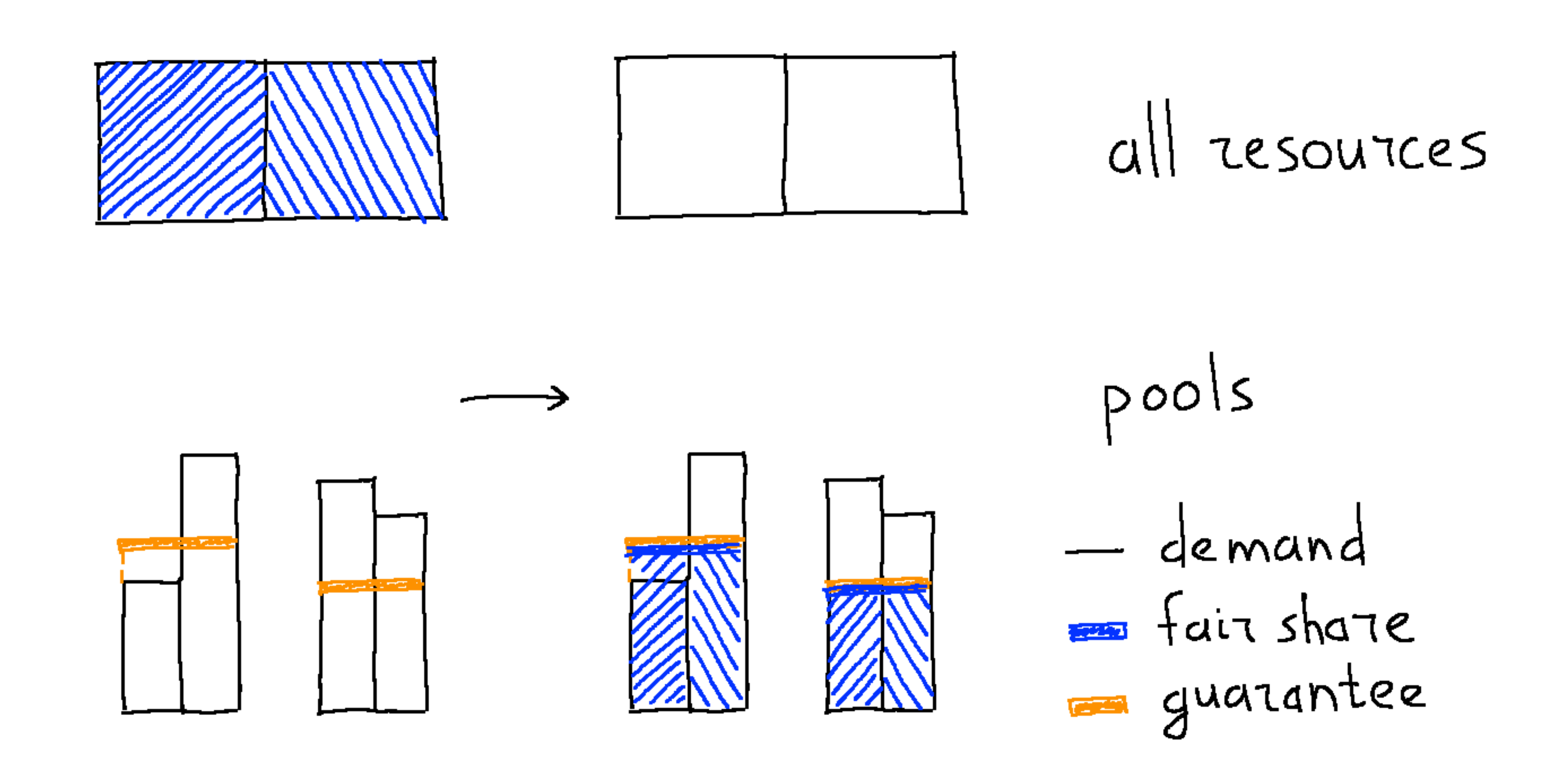
Чего не хватает?

Хочется делить ресурсы **иерархически**



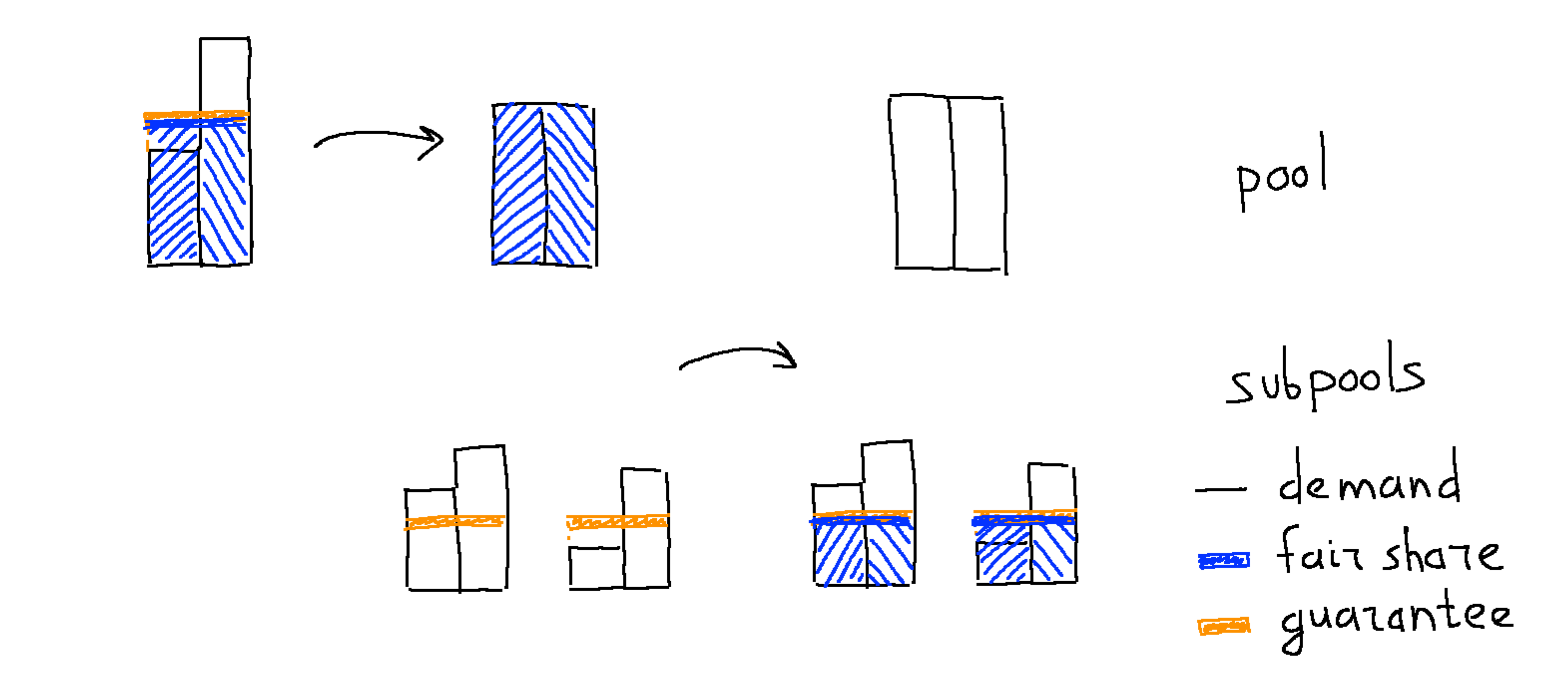
Hierarchical DRF

Делим ресурсы в корне



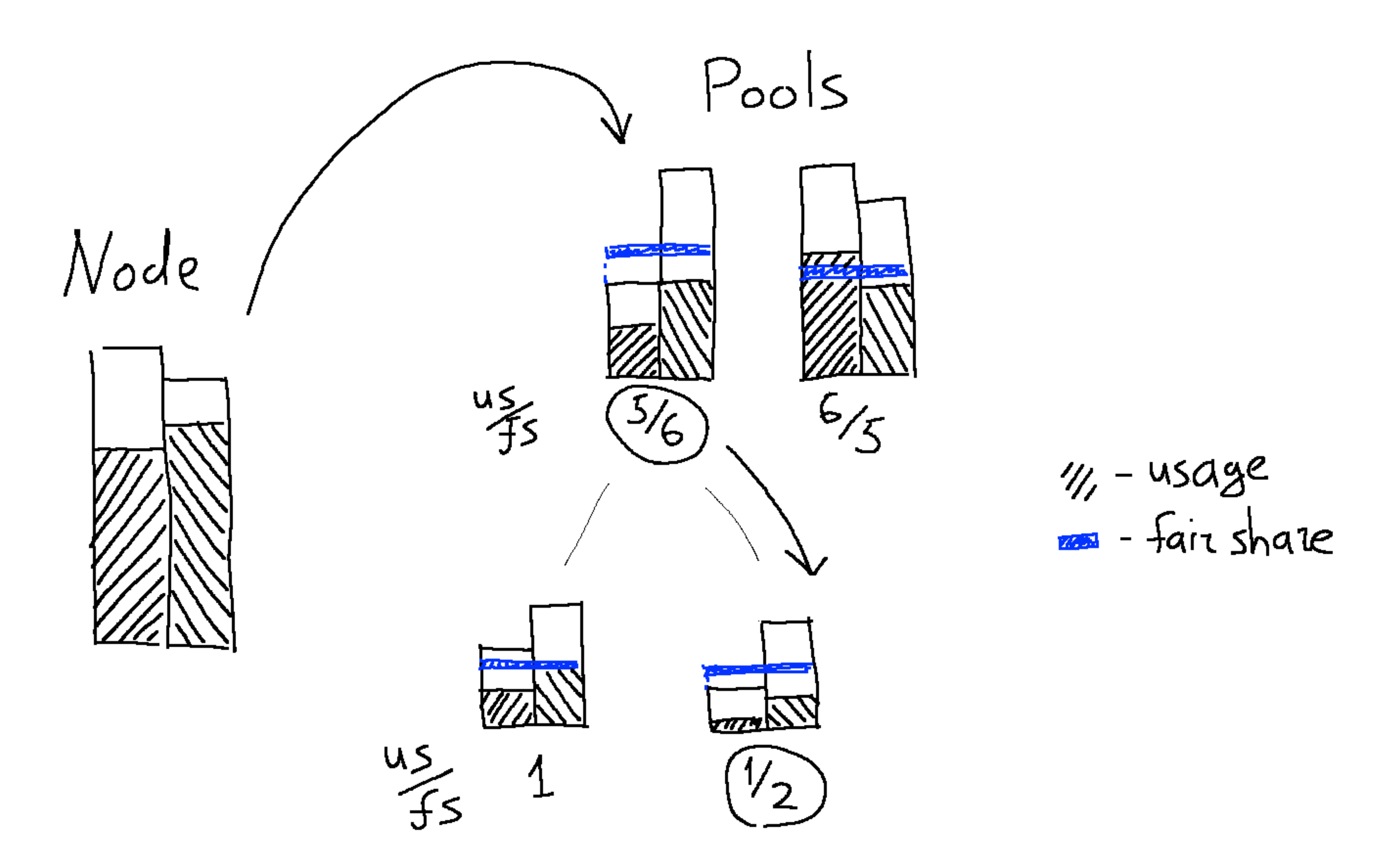
Hierarchical DRF

Делим ресурсы в подпулах

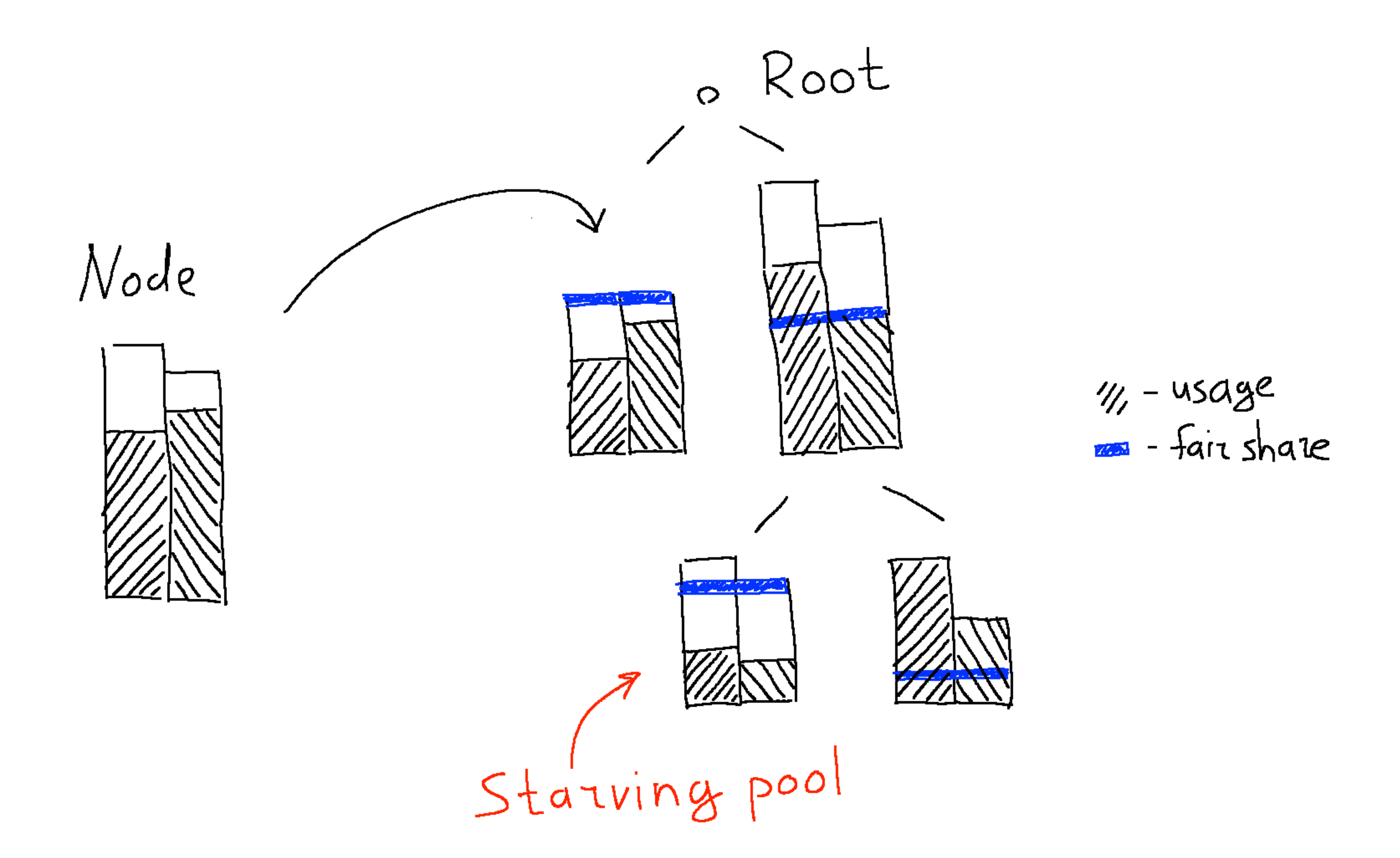


Hierarchical DRF: heartbeat

- Выбираем пул с минимальным us_{pool}/fs_{pool} в корне
- Повторяем рекурсивно

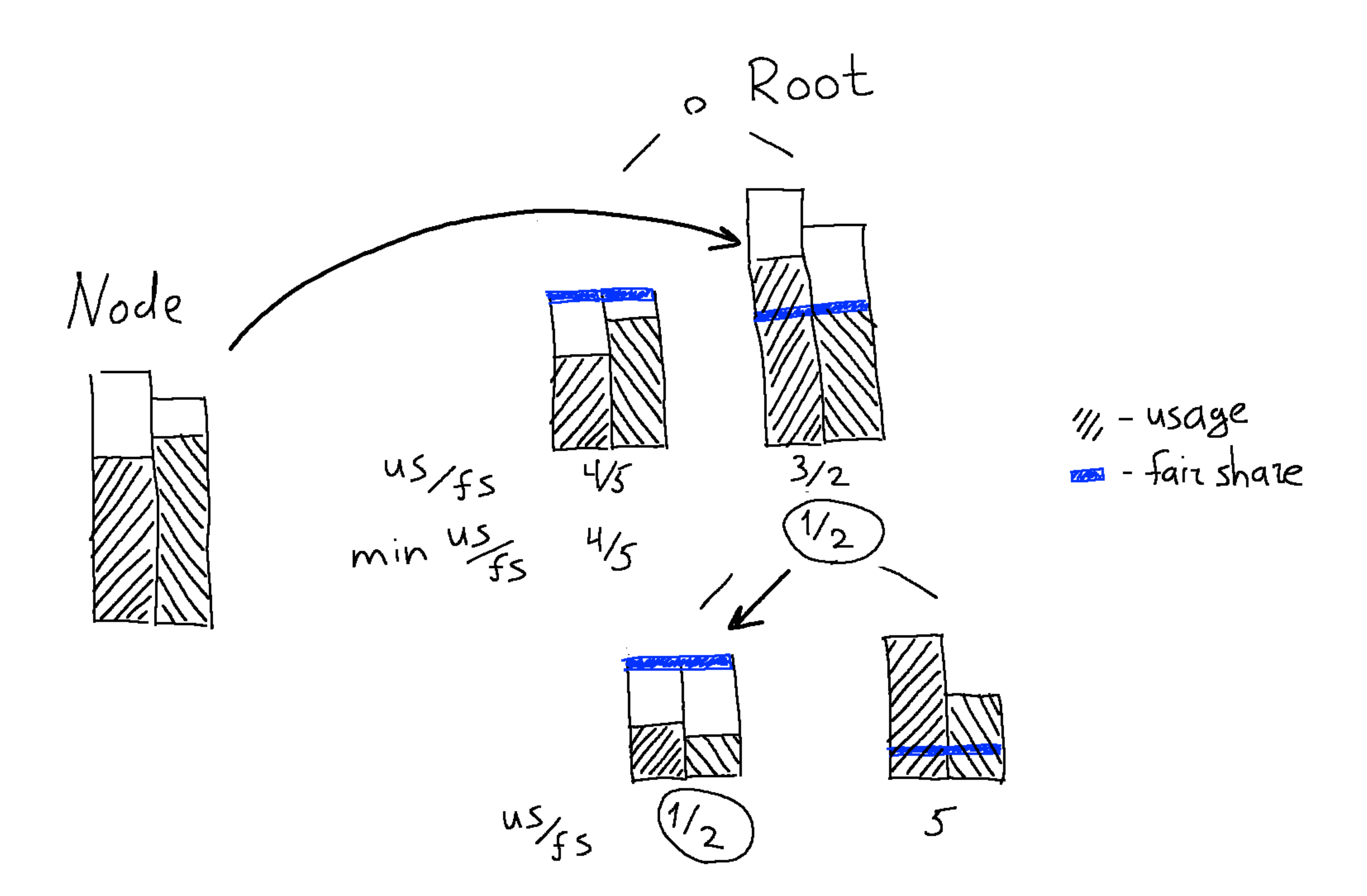


Hierarchical DRF: скрытая ~~угроза~~ проблема



Satisfaction HDRF: исправляем проблему

- Выберем поддерево с минимальным us_{pool}/fs_{pool}



Satisfaction HDRF

Есть неопубликованная статья, показывающая "корректность"

Hierarchical Fair Share Scheduling Revisited

Ignat Kolesnichenko   Andrey Kashin
Yandex LCC, Moscow, Russia   National Research University Higher School of Economics
ignat@yandex-team.ru   acid@yandex-team.ru

Abstract

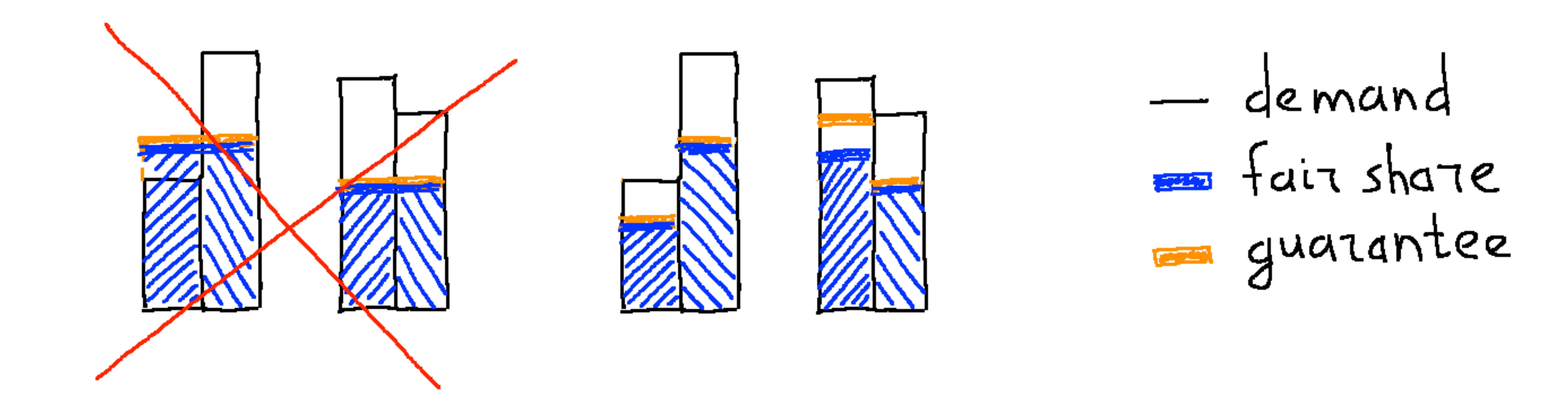
Problem of optimal and fair resource allocation in large heterogeneous systems has been actively researched during last decade. Moreover, the interest in efficient algorithms in this field still grows because even a 10 percent utilization improvement in cluster of 1000 machines saves a lot of resources and money for a company. The main goal of sched-

[?], Mesos [?]. These systems provide a lot of features for users to make their job easier. Among these features are: reliable and efficient storage, batch data processing, real-time key value storage. One of the challenges in designing such systems is providing efficient resource utilization of clusters along with some priority guarantees for tasks that run on such systems.

Чего еще не хватает?

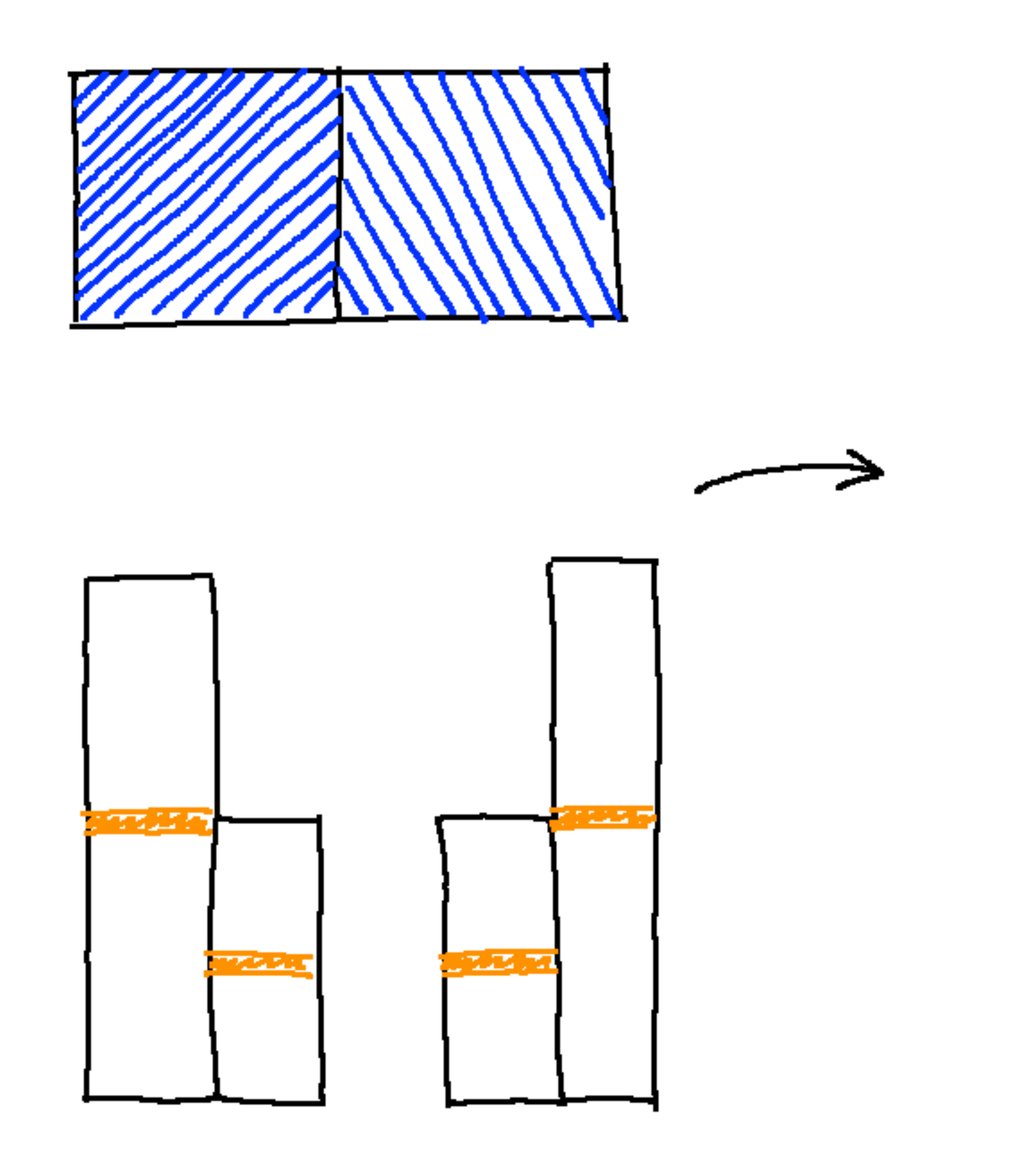
Хочется давать **векторные** гарантии

DRF умеет векторно раздавать ресурсы, но не умеет гарантировать



Vector HDRF: наивное решение

Давайте раздавать пропорционально гарантиям



all resources

pools

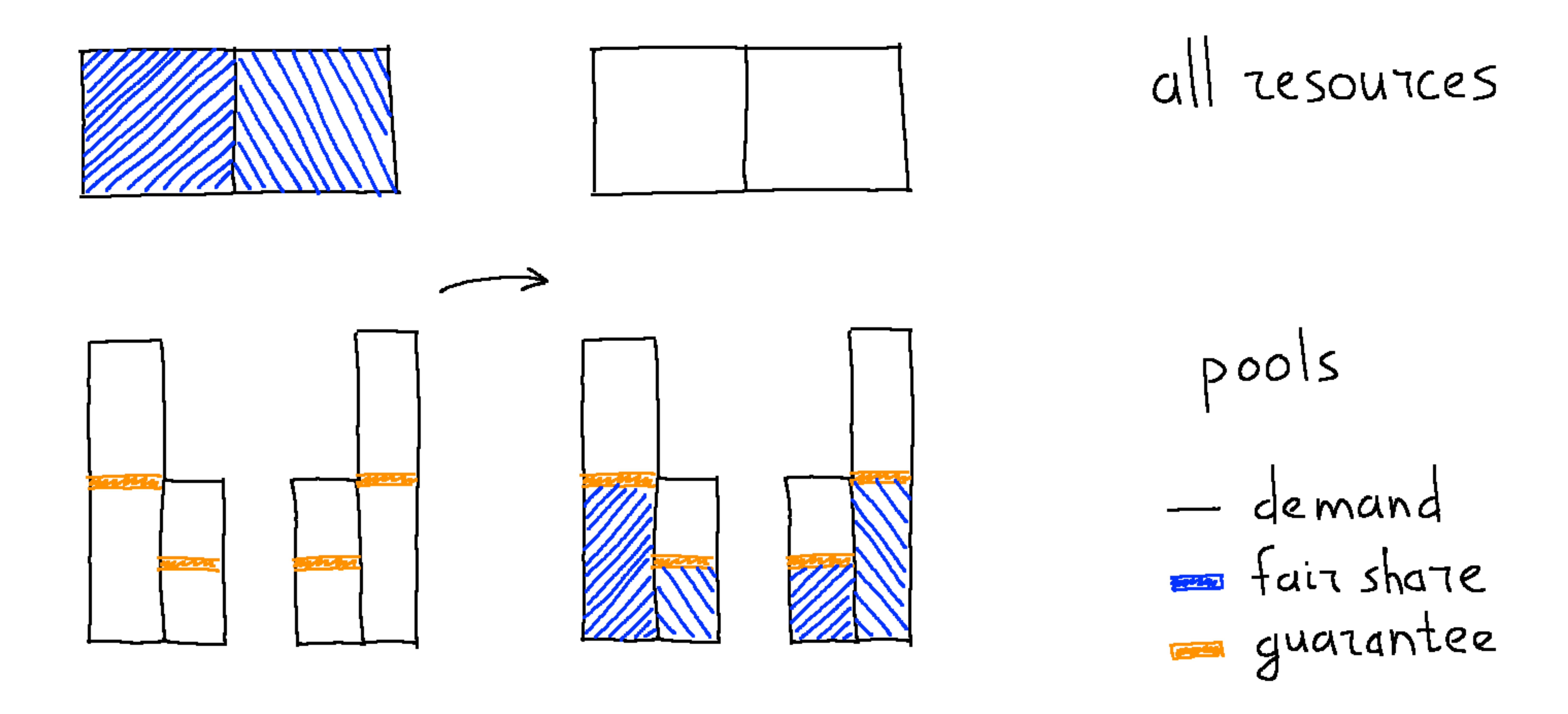
— demand

fair share

guarantee

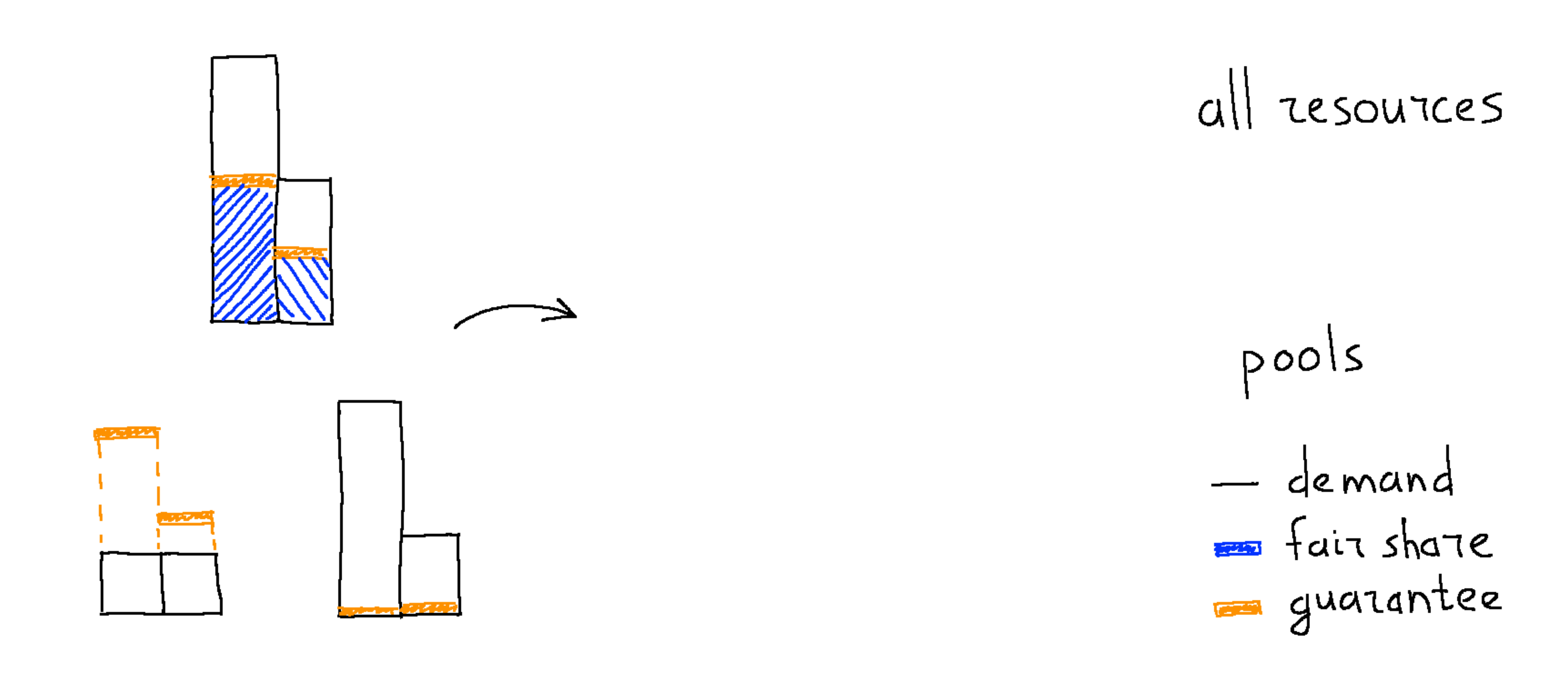
Vector HDRF: наивное решение

- Давайте раздавать пропорционально гарантиям



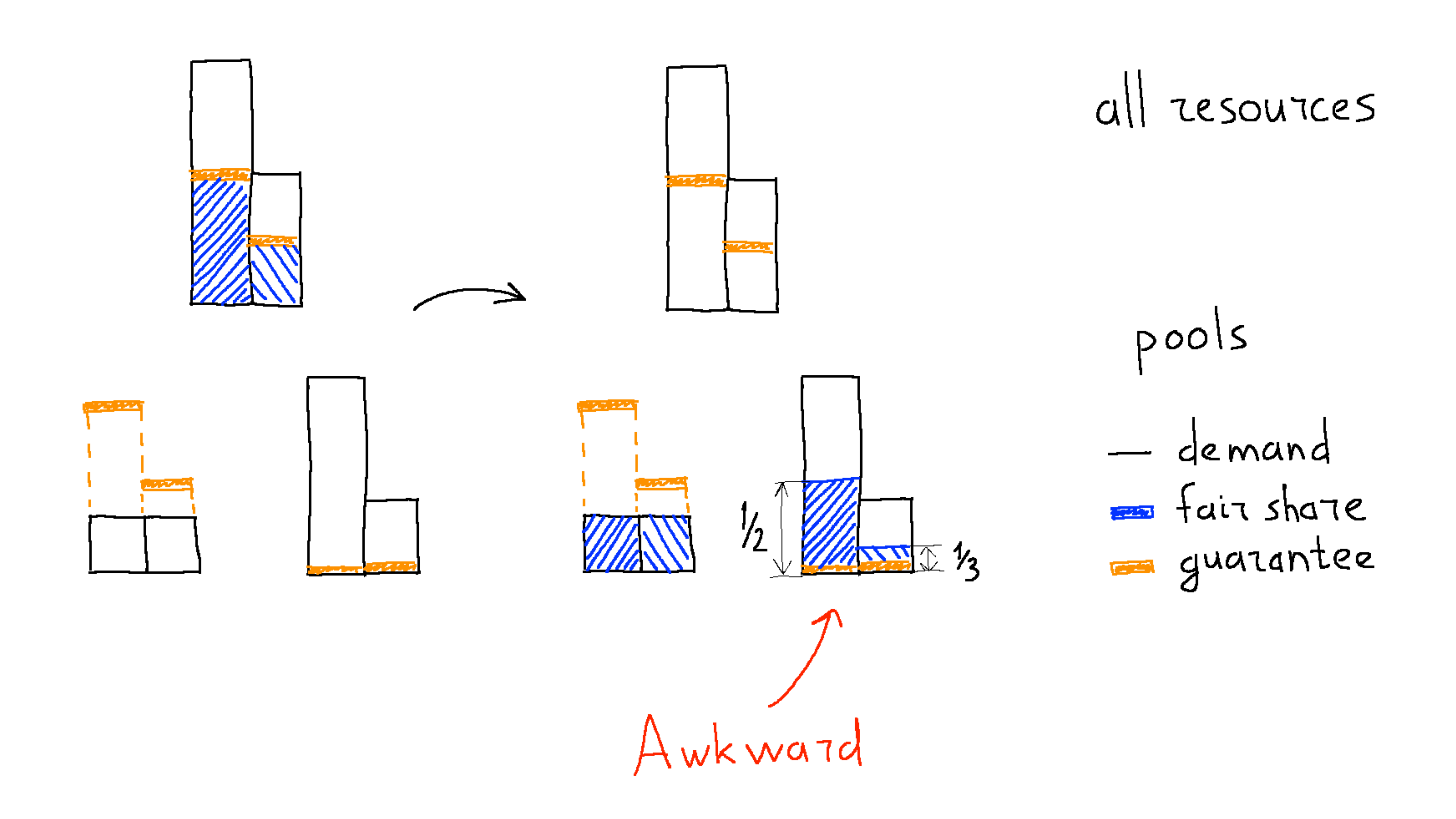
Vector HDRF: наивное решение

Давайте раздавать пропорционально гарантиям



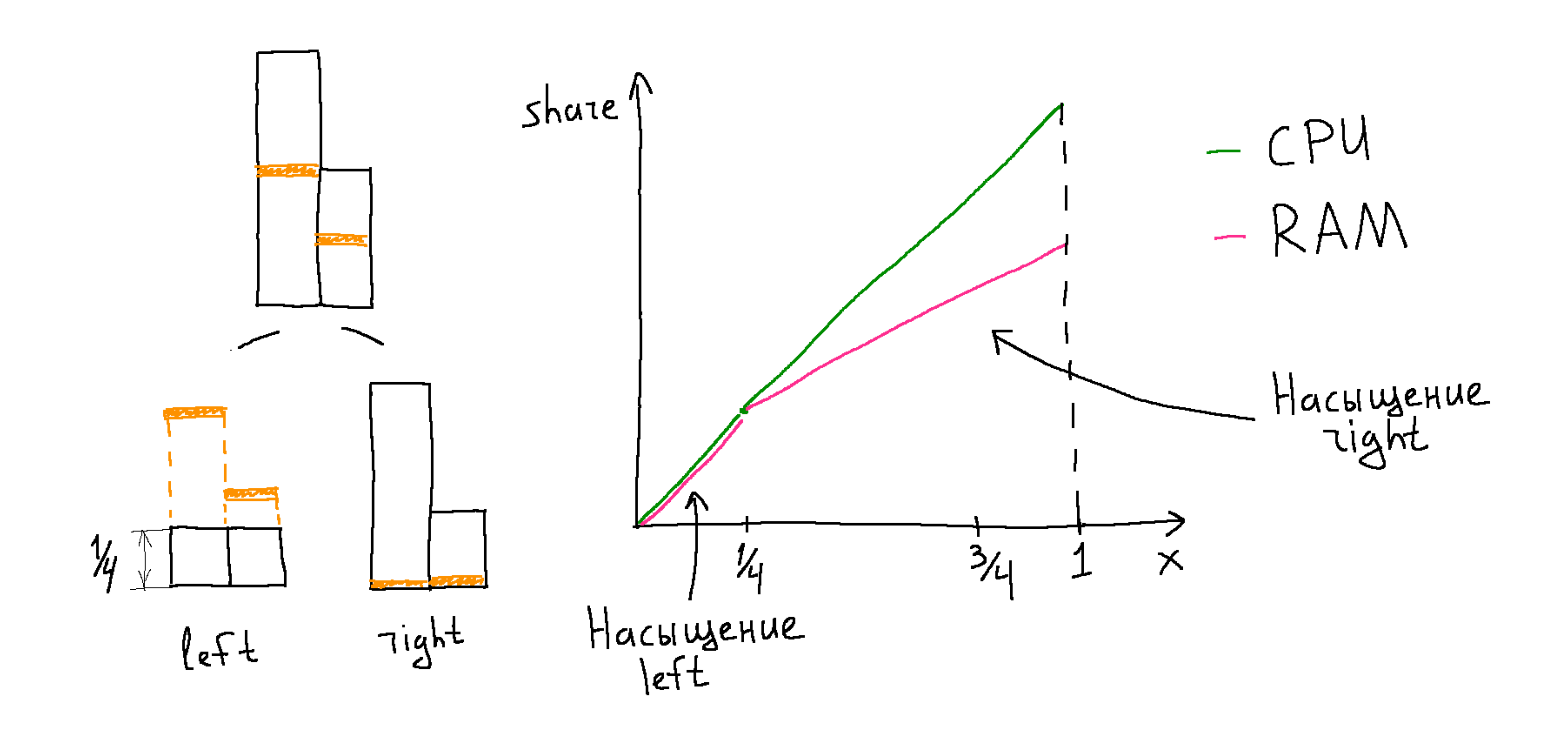
Vector HDRF: наивное решение

- Давайте раздавать пропорционально гарантиям



Vector HDRF

Построим для пулов функцию $suggestion(x) = \langle cpu(x), memory(x) \rangle$



Vector HDRF

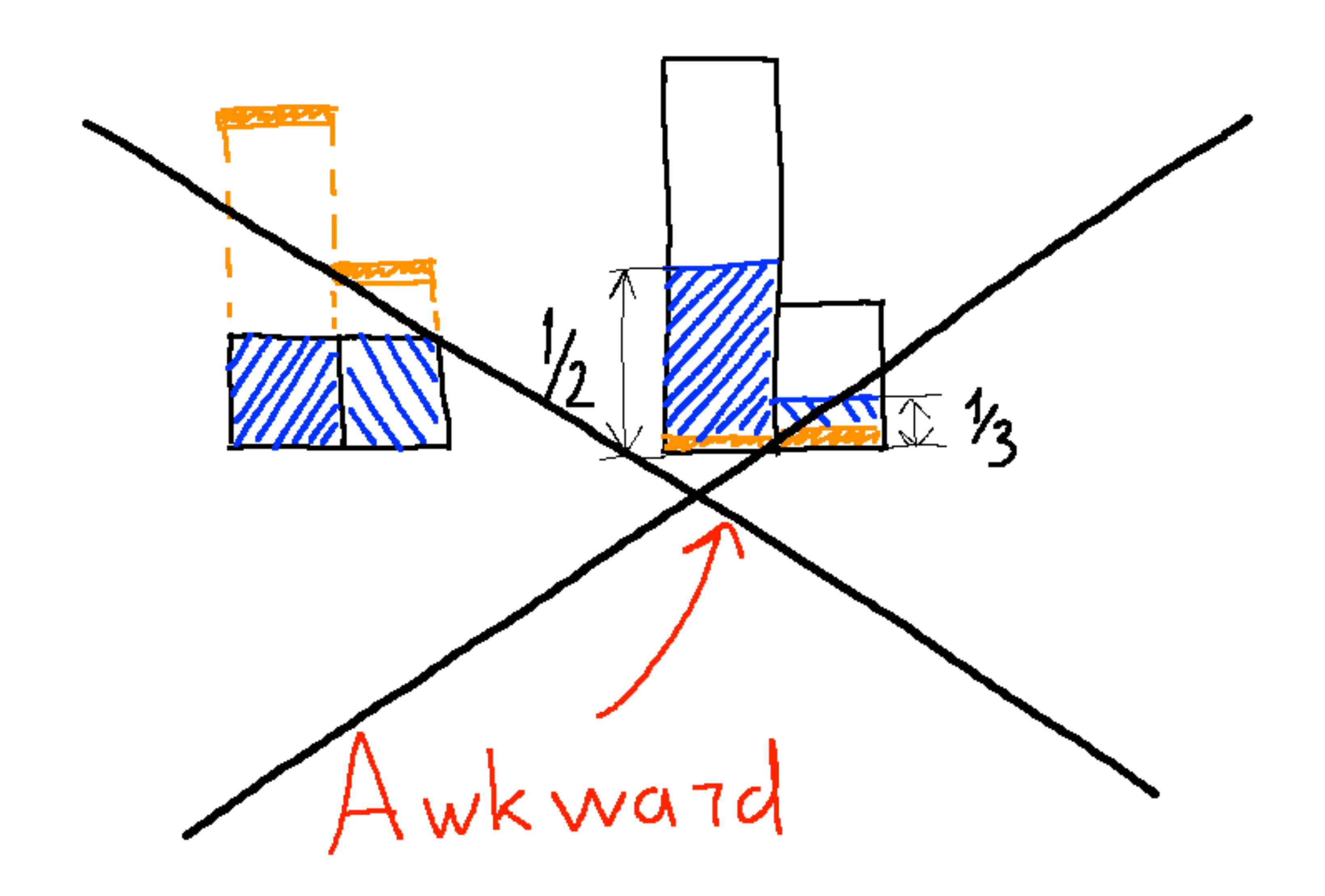
Построим для пулов функцию $suggestion(x) = \langle cpu(x), memory(x) \rangle$

- x – доля ресурсов кластера
- Функция кусочно-линейная

Может быть вычислена за линейное время от числа точек изгиба

Избегаем диспропорции demand и fair share

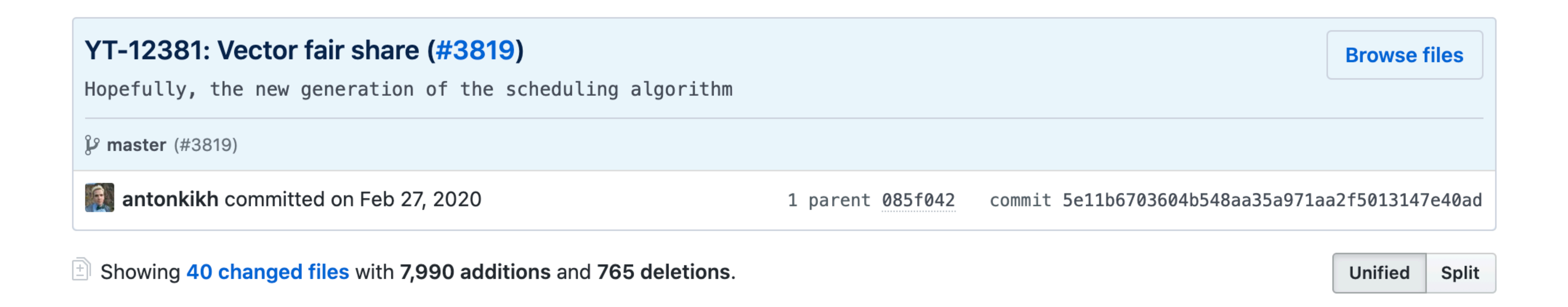
Детали опубликуем в статье



Vector HDRF

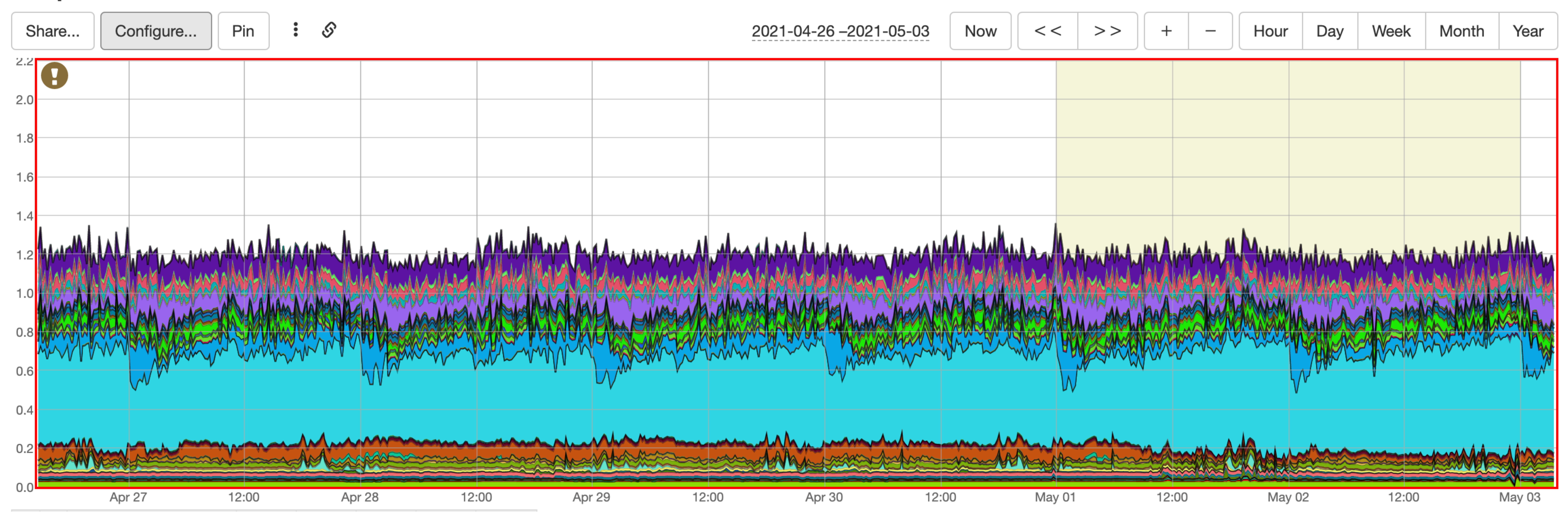
Реализован в YT

- Умеем выдавать непропорциональные гарантии
- Меньше проблем с вытеснением



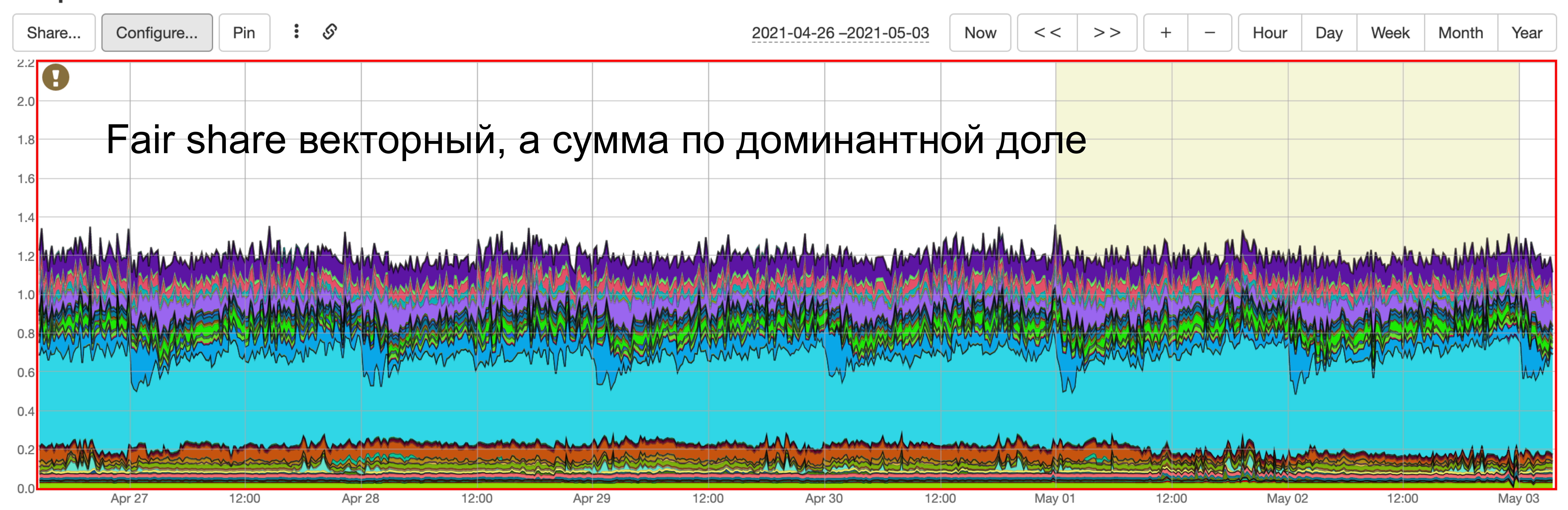
Vector HDRF

Так почему же тут сумма больше **единицы**?



Vector HDRF

Так почему же тут сумма больше **единицы**?



Выводы

Задача планирования – это непросто

- Необходимо планировать несколько ресурсов: CPU, RAM
- Необходимы пулы со строгими гарантиями
- Нужно поддерживать иерархию пулов

Что еще умеет планировщик YT

Что еще умеет планировщик YT

Обеспечивает интегральные гарантии

Что еще умеет планировщик YT

- Обеспечивает интегральные гарантии
- Поддерживает планирование на GPU

Что еще умеет планировщик YT

- Обеспечивает интегральные гарантии
- Поддерживает планирование на GPU
- Умеет на лету изменять гарантии

Что еще умеет планировщик YT

- Обеспечивает интегральные гарантии
- Поддерживает планирование на GPU
- Умеет на лету изменять гарантии
- Учитывает фактическое потребление ресурсов джобами

Яндекс

Всем спасибо

Колесниченко Игнат

Руководитель службы разработки планировщика

ignat@yandex-team.ru